Technische Universiteit Eindhoven University of Technology
Department of Mathematics and Computer Science
Data Mining Group

Window-Based Latent Representations for Multivariate Time Series Fault Classification

Master Thesis

I.Erdem Demir

Supervisors:
Sheng-Yi Hsu, SW-DDP-DATA-Predictive Maintenance & Diagnostics group, ASML, Development & Engineering
Professor Mykola Pechenizkiy, Eindhoven University of Technology

Final version

Eindhoven, August 2023

Abstract

The rapidly growing usage of machine sensors in industries has generated large volumes of multivariate time series data. Effective analysis of this data can lead to advancements in process optimization, fault detection, and predictive maintenance. Given the high dimensionality and potential noise in this data, we find it essential to use the latent representation approach to reduce dimensionality, extract features, and improve classification performance. This approach is particularly significant because the choice of window size directly influences the quality of these representations and the precision of fault detection. With this understanding, our research emphasizes the role of windowing in latent representations and the performance of state-of-the-art (SOTA) and pre-trained classification models. This emphasis bridges the gap in the literature, as there are no studies specifically addressing the impact of windowing on the representation qualities of multivariate time series data. To capture the separability and cohesion features of representations, we introduce two novel centroid-based latent representation quality metrics. Using these metrics, we explore the window-based representation qualities generated by SOTA and pre-trained models. To provide visual validation of these quantitative metrics, we employ a dimensionality reduction technique to plot latent representations. Our research also includes an evaluation of the classification performances of SOTA and pre-trained classification models across three public datasets. From this evaluation, we highlight the robustness of the OmniAnomaly model and the potential usage of pre-trained Wav2Vec 2.0 and ResNet50 models for fault classification. Finally, our study reveals a notable trade-off relationship between representation quality and classification performance for distinct window sizes.

Acknowledgements

First, I would like to thank the EIT Digital Master School for the opportunity to study as a Data Science (DSC) student in two exceptional universities, KTH Royal Institute of Technology as my entry university and Eindhoven University of Technology as my exit university. Secondly, I would like to thank my supervisor at ASML, my supervisor at TU/e, and all my other colleagues from ASML who helped me either by word or deed for their guidance, supervision, and help. Furthermore, I want to thank all my friends for their continued support.

A heartfelt thanks go to my family. You supported me through the whole period of my life and gave me all the opportunities I have; I am truly thankful that I have such a supportive and thoughtful family!

Contents

Contents vii

List of Figures ix

List of Tables xi

1 Introduction 1
1.1 A Challenge for the Industry . . . 2
1.2 Problem Description . . . 3
1.3 Research Questions . . . 4
1.4 Research Approach . . . 4
1.5 The Structure of the Thesis . . . 4

2 Background 5
2.1 Foundations of Multivariate Time Series . . . 5
2.1.1 Multivariate Time Series . . . 5
2.1.2 Temporal Aspects of Relationships . . . 6
2.2 Multivariate Time Series Representation . . . 6
2.2.1 Latent Representation . . . 6
2.3 Multivariate Time Series Analysis . . . 7
2.4 Fault Detection Classification . . . 8
2.4.1 Fault Detection Classification Studies In The Literature . . . 8
2.5 Evaluation Metrics for Imbalanced Datasets: G-Mean Score, Recall, and Specificity 9
2.5.1 G-Mean Score . . . 10
2.5.2 Recall (Sensitivity) . . . 10
2.5.3 Specificity . . . 10
2.6 Variational AutoEncoders (VAEs) . . . 11
2.6.1 OmniAnomaly . . . 12
2.7 GANs (Generative Adversarial Networks) . . . 13
2.7.1 MAD-GAN . . . 14
2.8 Wav2Vec 2.0 . . . 15
2.9 ResNet50 . . . 15

3 Methodology 17
3.1 Dataset Descriptions . . . 17
3.1.1 Server Machine Dataset (SMD) . . . 18
3.1.2 Soil Moisture Active Passive Satellite (SMAP) . . . 18
3.1.3 Mars Science Laboratory Rover (MSL) . . . 18
3.2 Preprocessing Datasets . . . 18
3.2.1 Feature Selection for MSL and SMD Datasets . . . 19
3.2.2 Sampling for ResNet50 Representation Generation . . . 19

3.2.3 Sliding (Overlapping) Window Approach 20
3.3 Schematic Overview of Train and Test Protocols 20
3.4 Prototypical Networks 22
3.5 Centroid-Based Representation Quality Metrics 22
3.5.1 Picking t-SNE For Representation Visualization 22

4 Results and Discussion **25**
4.1 Experiment SOTA Classification 25
4.1.1 Results For Experiment SOTA Classification 26
4.1.2 Results For OmniAnomaly 26
4.1.3 Results For MAD-GAN 28
4.1.4 Discussion For Experiment SOTA Classification 28
4.2 Experiment Pre-trained Classification 30
4.2.1 Results For Experiment Pre-trained Classification 31
4.2.2 Discussion For Experiment Pre-trained Classification 32
4.3 Experiment Representation Quality Evaluation 33
4.3.1 Results For Experiment Representation Quality Evaluation 34
4.3.2 Discussion For Experiment Representation Quality Evaluation 36
4.3.3 Visually Evaluating Representation Qualities 38

5 Conclusions **43**
5.1 Limitations and Future Work 44
5.1.1 Limitations 44
5.1.2 Future Work 44

Bibliography **45**

Appendix **49**

A Appendix **49**

List of Figures

2.1 The Block Diagram of a Variational AutoEncoder 11
2.2 The Detailed Architecture of OmniAnomaly Model [52] 12
2.3 The Structure of OmniAnomaly 13
2.4 MAD-GAN: Unsupervised GAN-based anomaly detection architecture 14
2.5 The Feature Encoder Architecture of Wav2Vec 2.0 15
2.6 The Architecture of ResNet50 Model 16

3.1 An Example Heatmap of A Window (Sized 4) Fed into Pre-trained ResNet50 Model to Generate A Representation 19
3.2 Schematic Overview of Train and Test Protocols For SOTA Models 20
3.3 Schematic Overview of Train and Test Protocols For Wav2Vec 2.0 Model 21
3.4 Schematic Overview of Train and Test Protocols For ResNet50 Model 21

4.1 OmniAnomaly and MAD-GAN Models' Experiment SOTA Classification Results . 29
4.2 Training Loss Curve Examples For SOTA Classification Models 29
4.3 G-Mean Scores Derived Using Prototypical Network Approach For Pre-trained Models 32
4.4 Comparison of Window-Based Representation Qualities for Wav2Vec 2.0, ResNet50, OmniAnomaly, and MAD-GAN Models ((0) Indicates Non-anomaly Cluster, (1) Indicates Anomaly Cluster) 35
4.5 Minimum of Inter-Centroid Distances Metric Comparison for All Models 37
4.6 Maximum Standard Deviation of Intra-Centroid Distances Metric Comparison for All Models 38
4.7 t-SNE Plot for Latent Representations of OmniAnomaly Model on MSL Dataset with Window Size 4 40
4.8 t-SNE Plot for Latent Representations of OmniAnomaly Model on MSL Dataset with Window Size 600 40
4.9 t-SNE Plot for Latent Representations of OmniAnomaly Model on SMAP Dataset with Window Size 4 41
4.10 t-SNE Plot for Latent Representations of OmniAnomaly Model on SMAP Dataset with Window Size 600 41
4.11 t-SNE Plot for Latent Representations of OmniAnomaly Model on SMD Dataset with Window Size 4 42
4.12 t-SNE Plot for Latent Representations of OmniAnomaly Model on SMD Dataset with Window Size 600 42

A.1 OmniAnomaly Model's Approximated Experiment Results Ran Three Times With Different Seeds 49
A.2 MAD-GAN Model's Approximated Experiment Results Ran Three Times With Different Seeds 49

List of Tables

2.1 Literature Review Comparisons 9

3.1 The Information of Three Public Datasets Used in Our Experiments 17

4.1 OmniAnomaly Experiment SOTA Classification Results 27
4.2 MAD-GAN Experiment SOTA Classification Results 28
4.3 Experiment Pre-trained Classification Results For Pre-trained Models 31

Chapter 1

Introduction

In the era of Industry 4.0, also referred to as the fourth industrial revolution, the integration of the Internet of Things (IoT) technology has emerged as a prominent catalyst for change [30]. This has facilitated data exchange across many industries and motivated the generation of an extensive volume of time series data. These data have the potential to improve operational efficiency, such as optimizing production lines and reducing waste, and enhance decision-making processes by providing real-time information on system performance and potential faults [55]. As industries strive to exploit the power of these data, the demand for advanced AI solutions and methodologies intensifies. In the context of Industry 4.0, such AI solutions present both challenges and opportunities [31].

In response to the growing demand for advanced AI solutions, the demand for sophisticated methodologies for time series data has also increased. Traditional techniques such as Fourier transforms and linear regression models have often faced challenges with the high dimensionality, noise, and complexities inherent in multivariate time series data [36]. Recognizing these challenges, we investigated the utilization of window-based latent representations for multivariate time series classification.

Latent representations, derived from methods such as auto-encoders or Gaussian processes, can reduce dimensionality, filter out noise, and extract meaningful features, which are crucial for accurate classification [45]. To further enhance our understanding of the dynamics of the data, we adopted the sliding window approach. This method, by capturing sequential data points in overlapping or non-overlapping windows, ensures that temporal dependencies and patterns, potentially overlooked in a single snapshot, are considered [34]. Using this approach, we aim to maintain and capture the temporal dependencies between different variables within the series.

With a particular emphasis on fault detection, our research seeks to address the limitations of linear and frequency-domain techniques by exploiting the use of latent representations. Through this endeavor, we would like to shed light on the complexities of multivariate time series data and contribute to the ongoing evolution of Industry 4.0.

The introduction consists of five sections. The first section introduces the challenge of fault detection classification using multivariate time series data and why there is a need for a stable multivariate time series representation model in the industry. The second section describes the problem and motivates our research. The third section formulates the identified research questions. The following section discusses the research approaches that would be used to answer the research question. The final section introduces the structure of the thesis.[1]

1.1 A Challenge for the Industry

Time series fault detection has a wide range of applications in the industry. With the help of fault detection, companies can monitor their key metrics and get quick insights into incidents on time. For instance, in SpaceOps and predictive maintenance, time series fault detection usage is several real-life scenarios in which time series fault detection is utilized. Also, it is used in various industries ranging from software applications, manufacturing, spacecraft, and servers to health care.

In the healthcare industry, quality and precision are highly important. Accordingly, the Siemens Healthineers team uses multivariate fault detection in the final inspection stage of their production. With the help of fault detection, the procedure used in the stress test of medical devices can be used to check the quality of the final product [3].

Moreover, in the aerospace industry, Airbus has developed a fault detection tool for its cognitive services to capture potential problems before they occur. When using aircraft monitoring applications and analyzing telemetry data from multiple flights, fault detection can be used to anticipate potential that might occur before flights. According to Marcel Rummens, Product Owner of the Internal AI platform, Airbus saved up to three months in developing smaller use cases with a fault detector [3].

In the semiconductor industry, ASML, a leading vendor of lithography systems, provides innovative solutions integral to producing integrated circuits, or chips, which are the core of electronic devices in the tech industry. ASML places a high priority on the topic of multivariate time series representation because the time series data collected by machine sensors, which are used in manufacturing processes, tend to be multivariate. Within the predictive maintenance department, fault detection applications can be used to proactively identify and resolve potential problems with complex machinery used in chip production. Therefore, fault detection solutions are vital to allow a more comprehensive understanding of machine behavior and the detection of earlier faults that might occur during production [9].

The careful and precise fault detection using multivariate time series data can improve the accuracy of predicting potential failures, minimize downtime, and enhance overall operational efficiency. Considering that the specific focus of the impact of windowing on fault classification and representation qualities has not yet been investigated, exploring this topic contributes to academic research and industrial applications. Furthermore, as explained above, studying this topic has considerable practical implications for many companies in their ongoing efforts to optimize operations.

[1]This master's thesis is the concluding part of the double-degree master's program - EIT Digital Master School in Data Science [1]. It was performed during the internship at ASML from March to August 2023 alongside the Data Mining (Data and AI Cluster) group at Eindhoven University of Technology.

1.2 Problem Description

On occasions such as synchronized sensor readings or coordinated activities occurring at the same time, time series data similarity could be effortlessly computed by calculating the Euclidean distance between corresponding points of two-time series that coincide. In other words, we can use the Euclidean Distance method for similarity evaluation when both time series are synchronous and uniformly progress. However, when the time series are out of sync, this approach quickly falls apart. For instance, when two sensors capture a machine's vibration, if one starts a few seconds later than the other, the Euclidean Distance method will misinterpret this delay as a significant difference in their readings, even though they are recording the same event. To address such challenges, techniques like Dynamic Time Warping (DTW) have been developed. DTW is a widely used technique for measuring similarity in time series data by aligning sequences, making it effective for tasks where temporal shifts exist [11]. However, as the input data grows, the computational intensity of DTW becomes a bottleneck. Here, deep learning models that provide a scalable solution can be used to generate latent representations. These representations can capture intricate patterns in multivariate time series, are robust to noise, allow for end-to-end training and transfer learning, reduce data dimensionality in an efficient manner, which helps overcome some of the difficulties associated with DTW in large and complex datasets. [45]. Our research addresses these challenges in time series data analysis.

Time series data involves a crucial "time" element that must be maintained, adding a layer of complexity to the representation process. In our research, we propose a novel perspective that centers on the role of windowing in shaping latent representations and its implications on the performance of state-of-the-art (SOTA) deep learning classification models. We use OmniAnomaly and MAD-GAN as SOTA models in the scope of our study, and Wav2Vec 2.0 and ResNet50 as pre-trained models (see model details in Chapter 2). By adopting a window-based evaluation perspective, we aim to maintain temporal dependencies and capture relationships among features. To investigate the impact of windowing, we chose five distinct window sizes (4, 40, 100, 400, and 600) for our experiments. Moreover, our goal is to compare the full time series classification results and representation qualities of the aforementioned deep learning models by retraining and testing them (representations generated during the testing process) with distinct window sizes on three public datasets. However, we would like to note that the optimal window size may vary depending on the specific application and the complexity of the time series data.

Existing evaluation techniques, such as statistical tests and dimensionality reduction approaches such as t-SNE, can provide insights, but do not fully capture the complexities of multivariate time series data [41]. Visualizations, while insightful, require quantitative support, especially for fault detection [57]. Our understanding is that representations from a specific class, whether anomalous or not, should cluster closely together. Nevertheless, representations from different classes should be distinctly separable, avoiding any blending. In line with this understanding, we introduce our novel centroid-based latent representation quality metrics. These metrics offer a more detailed perspective on the quality of latent representations, emphasizing both class cohesion and separability. Complementing this, we employ the t-SNE dimensionality reduction technique to visually interpret these representations in a 2D space. Together, these methodologies suggest that SOTA and pre-trained classification models have the potential to derive meaningful representations, offering a promising direction for further exploration in multivariate time series analysis.

In the following section, we present three research questions that guide our evaluations of classification performances and latent representation qualities of deep learning models.

1.3 Research Questions

- How does the window size parameter impact current state-of-the-art multivariate time series models' classification performance?
- How does the window size parameter impact pre-trained multivariate time series models' classification performance?
- What is the impact of window size on latent representation qualities? Can we evaluate latent representation qualities using dimensionality reduction techniques and quantitative evaluation metrics?

1.4 Research Approach

In advancing industry sustainability goals, the use of pre-trained models and transfer learning becomes essential in industrial applications. Using pre-trained models not only improves efficiency, but also promotes sustainable practices. Considering the sustainability point of view, reducing computational cost, working with limited resources, and not creating extra energy pollution while training large deep learning models, we opted for using pre-trained models of Wav2Vec 2.0 and ResNet50 in our research.

In addition to the pre-trained models, we propose a thorough analysis that aims to trace the windowing impact on classification performances and latent representation qualities of current SOTA classification models namely OmniAnomaly and MAD-GAN.

1.5 The Structure of the Thesis

To summarize, this study makes the following contributions:

1. Provides a thorough windowing impact evaluation of current SOTA multivariate time series classification models and pre-trained models for fault detection and classification.
2. Demonstrates the window size trade-off relationship between latent representation quality and classification performances of SOTA classification models and pre-trained models through evaluation of classification performances generated for different window sizes and latent representation quality comparisons.
3. Contributes to predictive maintenance and fault detection by introducing centroid-based latent representation quality metrics to evaluate representations. Also, using dimensionality reduction visualizations on representations allows one to visually comprehend the intuition behind those metrics and compare various representations generated using different models for distinct window sizes.

We explain these contributions in Chapter 4 in detail. By addressing the challenges and exploring innovative solutions, this study seeks to bridge the gap in multivariate time series data analysis and to set a precedent for future research in this domain.

Chapter 2

Background

In this thesis, we examine the effect of windowing on various SOTA multivariate time series representation models. Later, we investigated the latent representation qualities utilizing centroid-based representation quality measure metrics and pursued an efficient multidimensional representation using dimensional reduction techniques. Understanding the background of our experiments is crucial to comprehend our research while reading this thesis.

2.1 Foundations of Multivariate Time Series

2.1.1 Multivariate Time Series

Multivariate time series refers to a series of data points, typically measured over successive points in time, where multiple variables are observed at each time point [14]. It is a generalization of univariate time series where only one variable is observed at each time point.

$$X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{bmatrix} \tag{2.1}$$

Where:

- $X(t)$ is the multivariate time series at time t.
- $x_i(t)$ is the ith variable's value at time t.
- p is the total number of variables.

Analyzing multivariate time series data is crucial in various fields such as finance, economics, manufacturing, and healthcare, as it helps to understand the relationships and dependencies between different variables over time [14]. This form of analysis allows researchers to capture intricate patterns, trends, and cyclic behaviors that could be overlooked in univariate time series. When multiple variables are considered simultaneously, hidden correlations can be revealed, future trends can be predicted with greater precision, and meaningful insights can be derived. These insights are especially profound when recognizing temporal relationships within multivariate time series.

2.1.2 Temporal Aspects of Relationships

In multivariate time series, the temporal aspect of relationships refers to understanding how variables interact with each other over time.

$$\tau_{ij} = \text{correlation}\left(x_i(t), x_j(t+\delta)\right) \tag{2.2}$$

Where:

- τ_{ij} is the temporal relationship between the ith and j^{th} variables.
- δ is the time lag.
- correlation is a function that measures the relationship between two time series.

This involves not only capturing dependencies between variables at different time lags (and time intervals), but also underlying trends, seasonality, cyclic patterns, and auto-correlation of a variable with its past values [16]. Recognizing these temporal features is crucial for tasks such as forecasting and fault detection [16]. These temporal dynamics can be captured using the latent representation approach.

2.2 Multivariate Time Series Representation

The process of time series representation is of great importance for industries that aim to enhance their operations and employ various techniques for predictive maintenance, fault detection, and anomaly detection applications. Current SOTA (state-of-the-art time-series) representation approaches are efficient and commonly used for research purposes and aforementioned industrial applications. However, current approaches lack thorough research on how fault classification performance and their latent representation qualities of SOTA deep learning models are affected by the change in window sizes. Therefore, the impact of windowing has not yet been directly addressed in this context and its use in industry for fault detection.

Various methods have been proposed for the representation of multivariate time series data; one of the representation methods is feature-based representation, which involves extracting features that capture the underlying structure of time series data [37]. These features contain statistical measures like mean, variance, auto-correlation, or domain-specific features. Although using feature-based representation can reduce data dimensionality and highlight important characteristics, selecting features occasionally requires domain knowledge. Moreover, created features may not consistently capture the complex, non-linear relationships in multivariate time series data [17].

In contrast to feature-based representation, latent representation presents a more sophisticated approach to representing multivariate time series data.

2.2.1 Latent Representation

Latent representation involves converting multivariate time series data into a lower-dimensional latent space that captures the underlying structure of the data. Techniques such as dimensionality reduction methods or autoencoders can achieve this. The resulting latent representation can effectively reduce the dimensionality of the data and capture complex patterns, making it particularly suitable for tasks such as fault detection [59].

$$L(X) = f(X) \tag{2.3}$$

Where:

- $L(X)$ is the latent representation of the time series X.

- f is a transformation function that maps the original data to its latent representation.

Latent representation offers several advantages for fault detection:

- Effectively solves the high-dimensionality problem often associated with multivariate time series data. By transforming the data into a lower-dimensional space, latent representation techniques make the data more manageable and computationally efficient for fault detection [19].
- Latent representation can capture complex, non-linear patterns that may not be apparent in the raw data or through simple feature extraction. This ability to capture complex patterns is particularly important for fault detection, where faults often manifest as subtle deviations from normal patterns [58].
- Latent representation techniques, particularly autoencoders, are robust to noise in the data. This is crucial for fault detection, as noise can often obscure the presence of anomalies [17].
- Latent representation techniques are unsupervised, meaning that they do not require labeled data. This is beneficial for fault detection, where labeled data are often scarce [59].
- Although the transformed data in latent space is difficult to interpret directly, the transformation process can provide insight into the underlying structure of the data. For instance, the principal components in PCA represent directions of maximum variance in the data, while hidden units in an autoencoder can capture salient features in the data [19].

Although feature-based representation methods for representing multivariate time series data have advantages, latent representation offers several clear advantages, particularly for complex tasks like fault detection. By effectively reducing dimensionality, capturing complex patterns, providing robustness to noise, and offering unsupervised learning capabilities, latent representation emerges as a powerful tool for fault detection in multivariate time series data [59].

2.3 Multivariate Time Series Analysis

In the field of data science, time series analysis is a crucial component, particularly in disciplines where temporal patterns and trends hold significant value. From detecting anomalies in sensor data to predicting stock market trends, the analysis of time series data can provide profound insights. However, the increased complexity of multivariate time series analysis requires custom or more advanced tools and methods compared to univariate time series analysis.

Packages such as tsfresh[18] and pyts[21] are commonly used in the context of multivariate time series analysis. To elaborate, tsfresh is a feature extraction package that primarily extracts a wide array of individual features from each time series independently [18]. Tsfresh can utilize its feature extraction methods to extract characteristics of features that can be used for regression and classification tasks. In our study, we focus on using the latent representation approach. However, the tsfresh package does not provide any methods that follow the latent representation approach [18].

Pyts is a Python package that specializes in time series transformation and classification [21]. Pyts offers a variety of transformation techniques such as Symbolic Aggregate Approximation (SAX), Continuous Wavelet Transform (CWT), and Discrete Fourier Transform (DFT). In this regard, these transformation methods can be used for several time series analyses, such as reducing the dimensionality of the time series data and converting them into a format suitable for traditional machine learning algorithms. However, using these methods may not fully capture the temporal dependencies and dynamics inherent in multivariate time series, which are often essential for effective fault detection.

On the other hand, as explained in Section 2.2, latent representations can capture key features of the time series while reducing the dimensionality of the data. Reducing the dimensionality using

representations minimizes the risk of overfitting and improving computational efficiency [38]. The implication of using the latent representation approach in the context of fault detection is that a deep learning model can accurately identify anomalous patterns indicative of a fault, even in the presence of noise and other complicating variables such as sensor drift, environmental fluctuations, transient spikes, or interference from other equipment [53, 32, 42]. As a result, using the latent representation approach for multivariate time series analysis offers interpretability for our study. In this context, we particularly investigate the effects of changing the size of windows in fault detection.

2.4 Fault Detection Classification

Fault Detection Classification focuses on detecting and classifying anomalies, often called "faults" in multivariate time series analysis. Faults can indicate errors, system malfunctions, and important events in industrial robotics, process control, and manufacturing. In addition to detecting anomalous or abnormal behavior on multivariate time series data, fault detection involves identifying deviations from expected patterns or the relation between multiple variables over time. Later on, identify instances where the system is not functioning as expected in the model.

In summary, Fault Detection Classification analyzes multivariate time series data that incorporate representation feature extraction, fault detection, and classification algorithms. This implies discovering and classifying faults within large and complicated data sets by extracting pertinent features, identifying abnormalities, and assigning fault labels. Fault Detection Classification has applications in many areas where the reliability and performance of systems depend on the accurate detection and classification of anomalies in multivariate time series data.

2.4.1 Fault Detection Classification Studies In The Literature

In the context of fault detection classification, we reviewed deep learning studies that utilize latent space representations of time series data.

Autoencoders, a specific form of artificial neural networks, are designed to learn compact and efficient encodings of data in an unsupervised manner. Their primary objective is to discover a representation or encoding for a dataset, often used for the purpose of dimensionality reduction. In recent times, autoencoders have been extensively utilized for learning probable distributions of input data and have found significant applications in anomaly detection. This is achieved by transforming the input data into a lower dimension and subsequently reconstructing it. A high reconstruction error is indicative of anomalous data [50].

Autoencoders can be utilized to generate representations that capture the changing temporal characteristics of the data in the context of multivariate time series classification. For instance, Fazle Karim et al. proposed a novel deep learning-based approach, DeTSEC, which uses an attentive-gated recurrent unit (GRU) based autoencoder to learn a new data representation. The model is trained in two stages: first, the autoencoder is trained to reconstruct the original data, and then a clustering refinement loss is introduced to further stretch representations towards the corresponding cluster structure. The authors showed that DeTSEC outperforms several competing methods on six real-world multivariate time series benchmarks, demonstrating the effectiveness of autoencoders in this context [22].

However, traditional auto-encoders might overfit the input data, which leads to poor generalization performance. This is particularly problematic when analyzing multivariate time series data, which often contains intricate temporal dependencies and noise. To address this issue, variational autoencoders (VAEs) have been suggested, introducing a probabilistic graphical model with an encoder, a decoder, and a loss function that allows the model to focus on learning meaningful representations in the presence of noise.

To solve temporal dependencies and the stochastic nature of the data, Su et al. [52] proposed a model named OmniAnomaly to address this issue. OmniAnomaly leverages a recurrent neural network (RNN) architecture, including a variational autoencoder (VAE) structure, to reconstruct the input multivariate time series data. The model is particularly effective at handling the temporal dependencies and stochastic nature of the data, making it a valuable tool for fault detection in multivariate time series [52]. However, by choice of the window size for the OmniAnomaly model, especially for the windows with a low ratio of anomalies, the model's performance may be affected, and thus becomes the subject of our research.

Another notable work in this field is the Multivariate Anomaly Detection with Generative Adversarial Networks (MAD-GAN) proposed by Li et al. MAD-GAN uses a generative adversarial network (GAN) to model the normal data distribution and detect anomalies. The model uses multiple generators to capture the multi-modality of normal patterns and a discriminator to differentiate between normal and abnormal patterns. The model was evaluated on multiple real-world datasets and demonstrated superior performance in anomaly detection compared to other state-of-the-art methods [35]. However, MAD-GAN's reliance on GANs may make it sensitive to the choice of window size, and it may struggle with high-dimensional data or multivariate time series data with complex dependencies.

The ResNet50 model, proposed by He et al.[26], is a deep residual network that uses shortcut connections to solve the problem of vanishing gradients in deep neural networks. Although primarily used for image classification tasks, its deep residual learning framework can be effectively applied to time-series data. The model's ability to learn complex representations makes it a valuable tool for time series analysis. Nevertheless, ResNet50's depth may make it computationally expensive and potentially prone to overfitting, especially when dealing with small or noisy datasets. [26].

The Wav2Vec 2.0 model, developed by Baevski et al.[10], uses self-supervised learning to learn speech representations. While primarily used for speech data, its approach to learning representations from raw audio data can be extended to time series data. The model's ability to learn meaningful representations from raw audio data makes it a powerful tool for time series analysis. [10] However, Wav2Vec 2.0's focus on audio data may limit its effectiveness in multivariate time series data from various domains.

Model	Deep Learning Architecture	Application	Input Data
OmniAnomaly[52]	GRU+VAE	Anomaly Detection	Multivariate Time-Series
MAD-GAN[35]	GANs	Anomaly Detection	Multivariate Time-Series
ResNet50[26]	CNN	Image Classification	Multivariate Time-Series
Wav2Vec 2.0[10]	Transformer+CNN	Speech Recognition	Univariate Time-Series

Table 2.1: Literature Review Comparisons

Last but not least, it is essential to highlight a particular challenge inherent in Fault Detection Classification: the imbalanced nature of the data used in its applications. In many systems, while there might be an abundance of normal data points, anomalies or faults are rare. This imbalance makes their detection and evaluation even more crucial, but also more challenging. Recognizing this, our study specifically adhered to this characteristic of imbalanced data and aims to identify instances where the system is not functioning as expected.

2.5 Evaluation Metrics for Imbalanced Datasets: G-Mean Score, Recall, and Specificity

Imbalanced datasets are a common challenge in machine learning, particularly in classification tasks. In cases where one class of data significantly outnumbers the other, the model shows a

tendency to be biased toward the majority class. The imbalance within the dataset can lead to misleading evaluation metrics, often for the class of interest. To address this issue, several evaluation metrics have been developed specifically for imbalanced datasets, including the G-Mean score, Recall, and Specificity.[60]

Before explaining the three evaluation metrics used for the imbalanced dataset, first, we would like to introduce the terminologies of true positive (TP), true negative (TN), false negative (FN), and false positive (FP) values.

TP and TN represent instances where the model correctly identifies the class, whether the correctly predicted label is positive or negative. On the other hand, FP and FN occur when the model incorrectly classifies the instances. These four terms form the foundation for calculating key performance metrics such as G-mean, recall, and specificity scores providing a comprehensive measure of the model's performance.[46]

2.5.1 G-Mean Score

The G-Mean or Geometric Mean is a metric that considers both recall (sensitivity) and specificity. It is particularly useful for imbalanced datasets because it gives equal weight to both classes, ensuring that the majority class does not overshadow the performance of the minority class.[60] The G-Mean is calculated as the square root of the product of sensitivity and specificity.

$$G_mean = \sqrt{recall \times specificity},$$

A high G-Mean score indicates that the model performs well in both classes, making it a valuable metric for evaluating models trained on imbalanced datasets.[60]

2.5.2 Recall (Sensitivity)

Recall, also known as sensitivity or true positive rate, is a measure of a model's ability to correctly identify positive instances from all actual positive instances. Recall is a crucial metric in the context of imbalanced datasets because it specifically focuses on the model's performance on the minority class. High recall indicates that the model correctly identifies a high proportion of the positive (minority) class, often the class of interest in imbalanced datasets.

$$recall = \frac{TP}{TP+FN}$$

2.5.3 Specificity

Specificity, also known as the true negative rate, measures the proportion of actual negatives that are correctly identified. In other words, it measures the model's ability to correctly identify negative instances from all actual negative instances. While recall focuses on the model's performance in the positive (minority) class, on the other hand, specificity focuses on the model's performance in the negative (majority) class. A high specificity indicates that the model correctly identifies a high proportion of the negative (majority) class.

$$specificity = \frac{TN}{TN+FP}$$

In conclusion, the G-Mean score, Recall, and Specificity are valuable metrics for evaluating machine learning models trained on imbalanced datasets. They provide a more nuanced understanding of a model's performance than traditional accuracy metrics, particularly on the minority class, which is often the class of interest in imbalanced datasets.

2.6 Variational AutoEncoders (VAEs)

Variational AutoEncoders (VAEs), unlike traditional AutoEncoders, use variational inferencing. It is a reconstruction-based deep learning anomaly detection model that can be used for robust anomaly detection [33]. VAEs offer a probabilistic approach to autoencoding, in which they encode the input data not into a fixed set of values but into a distribution. This generative characteristic of VAEs provides the model with the ability to generalize to new data that have never been seen before in the training dataset.

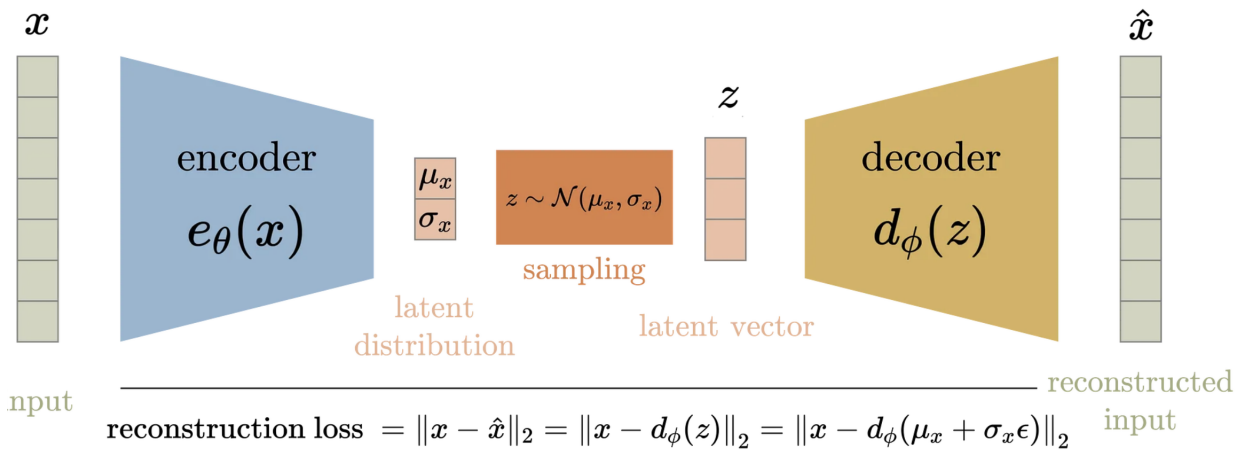


Figure 2.1: The Block Diagram of a Variational AutoEncoder

As shown in Figure 2.1, the VAE model consists of two main parts: an encoder (or recognition) network and a decoder (or generation) network. The encoder network models the input data as a distribution in the latent space to learn the underlying structure of the data. The decoder network then samples points from this latent space distribution to generate new data. The major distinction between a standard autoencoder and a VAE is that VAEs, by adapting a probabilistic graphical model on the latent space, allow sampling, thus the generation of new data. This is achieved through the reparametrization trick, which allows for backpropagation through the random nodes and forms the essence of VAEs [33].

Variational Inference (VI) [24] is a strategy in machine learning that approximates complex distributions using simpler, more tractable ones. Variational inference is essential for VAE. It is a method of approximating intractable integrals and is often applied in the context of Bayesian inference[51] to approximate posterior distributions. A typical autoencoder aims to learn a deterministic function that maps an input to a latent representation. However, VAEs, unlike standard autoencoders, introduce probabilistic latent variables and, thereby, a degree of randomness into the model. This added complexity requires the use of VI.

Assuming a complex posterior distribution $p(z|x)$ (where z is the latent variable and x is the observed data), it is not possible to directly calculate the posterior distribution. VI addresses this by introducing a simpler parametric distribution $q(z|x;\phi)$, typically Gaussian, controlled by parameters ϕ. This auxiliary distribution, also termed the variational distribution, is designed to approximate the true posterior.

Within the VAE framework, the encoder network parameterizes the variational distribution $q(z|x;\phi)$. It learns to represent the input data x as the distribution parameters in the latent space. Conventionally, this is a Gaussian distribution with mean μ and standard deviation σ, denoted as $q(z|x;\phi) = N(z;\mu,\sigma)$.

The objective then becomes the optimization of the parameters ϕ (of the encoder) and θ (of the decoder) to maximize the Evidence Lower Bound (ELBO)[24], a surrogate objective for the log-likelihood of the data. The ELBO comprises two terms: the reconstruction loss, a metric of how effectively the decoder reconstructs the original data from the latent representation, and the KL-divergence, known as the Kullback-Leibler (KL) divergence, between the approximated posterior and the prior, a regularizer term encouraging the encoder to produce latent variables following the prior distribution. KL-divergence forces the latent variables to follow a unit Gaussian distribution. This prevents the model from assigning too much importance to any specific input in the training data, and hence, ensures that the model generalizes well to unseen data.

While VAEs have shown remarkable success in various applications, one major downside is the so-called 'posterior collapse', where the model becomes overly reliant on the prior and ignores the latent variables. To prevent the 'posterior collapse', several methods are suggested, such as KL-Annealing [23], Normalizing Flows [44], ELBO Surgery[27], and so on. This has been addressed in OmniAnomaly[52] paper by proposing Normalizing Flows [44] in the encoder of a VAE to learn a more complex posterior distribution, with the aim of improving the performance of VAEs in certain specific scenarios [13].

2.6.1 OmniAnomaly

OmniAnomaly is an extension of VAE, which consists of a GRU layer to capture temporal dependencies between data points within time series windows and a VAE layer to reconstruct the input value to detect whether there is an anomaly within the provided input data.

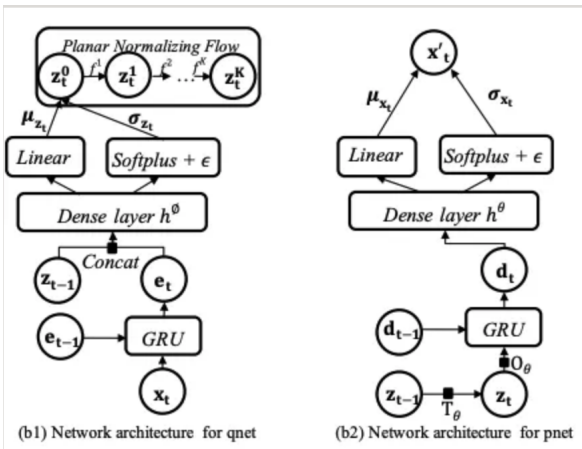


Figure 2.2: The Detailed Architecture of OmniAnomaly Model [52]

OmniAnomaly [52] consists of two main components which are online detection and offline training. The first step, data pre-processing, is shared by both modules, and it is used for standardizing and segmenting the data set into sequences using sliding windows. After pre-processing,

a training time series that covers a specific period is processed in the model training section, which learns from a model that captures normal patterns of the time series and calculates an anomaly score for each examination. Additionally, these anomaly scores have been used by the threshold selection part to automatically define anomaly thresholds with the Peaks Over Threshold (POT) method. Peaks-Over-Threshold (POT) is a theorem in Extreme Value Theory (EVT) that fits the tail of a probability distribution (anomalies locate at the low end of the distribution) using a generalized Pareto distribution (GPD) to identify extreme values without assuming a specific data distribution [47]. Furthermore, this offline training process can be compiled flexibly depending on data characteristics such as weekly or monthly.

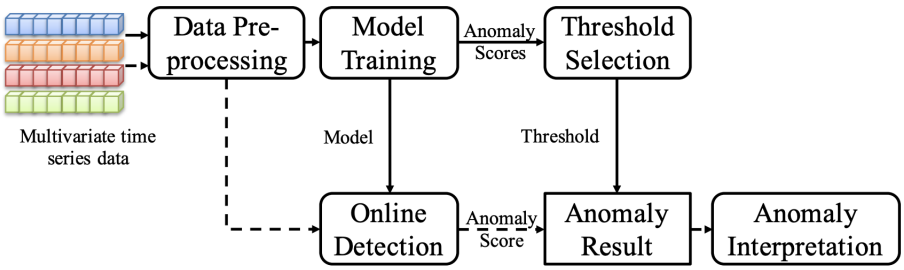


Figure 2.3: The Structure of OmniAnomaly

During the inferencing phase, the online detection part stores the trained model and receives observations indicated as x_t at time t after preprocessing. Later, all generated observations are passed to the online detection process, which is used to score the observations. As if the anomaly score is higher than the predetermined threshold, x_t is classified as an anomaly; otherwise, classified as normal. When there is an anomaly, the contribution or importance of each dimension error in x_t is calculated and ranked according to reconstruction probability. As a result, it allows for analysis of defining anomaly and presenting insights into the specific dimensions and variables that cause the anomaly.

2.7 GANs (Generative Adversarial Networks)

Goodfellow et al. [25] used a generative adversarial network to estimate generative models while using an adversarial process. The framework consists of training two main models simultaneously. The first model is a generative model (G) that focuses on the data distribution. Secondly, the discriminative model (D) is used to predict the probability of the sample acquired from the generative model or the training data. The purpose of training the generative model is to maximize the success rate of the discriminative model. By utilizing multilayer perceptrons such as G and D, the model is capable of trained backpropagation.

Despite significant success in discriminative models, deep generative models perform well due to contrary probabilistic computations and the hardness of leveraging the beneficial parts of piecewise linear units. However, the proposed generative adversarial networks solve this difficulty by setting the generative model to the discriminative model to have more satisfactory results.

In summary, the generative model is meant for producing human-like data that is indistinguishable from manually produced data; on the other hand, the discriminative model is set to detect false data. While these models were running simultaneously, they both improve their predictions until the generated data are indistinguishable from the real data.

2.7.1 MAD-GAN

In order to address abnormalities in complex datasets, Li et al. [35] have proposed "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks that uses Generative Adversarial Networks" (GANs 2.7).

In addition to previous approaches, this method is constructed to consider all variables together and detect the relations between them. By analyzing historical multivariate time series data, they aimed to detect real-time anomalies. During the implementation, both the GAN's discriminator and the generator were adapted for inspecting the abnormalities by assigning a unique anomaly score, which incorporates reconstruction residuals and discrimination results.

MAD-GAN employs Long-Short-Term Recurrent Neural Networks (LSTM-RNN), a form of neural network, to produce fake data and distinguish between actual and fake data. The generator uses random inputs to generate fake time series samples, while the discriminator works to distinguish between the fake samples and actual training data.

Instead of treating each variable separately, MAD-GAN adopts a different strategy and takes into account all of them simultaneously. Utilizing the sliding-window method, it breaks the time series down into smaller sub-sequences. This enables the model to capture varying interactions at various degrees of detail. The goal of training the generator and discriminator is to improve the discriminator's ability to distinguish between genuine and deceitful sequences, while the generator seeks to trick the discriminator.

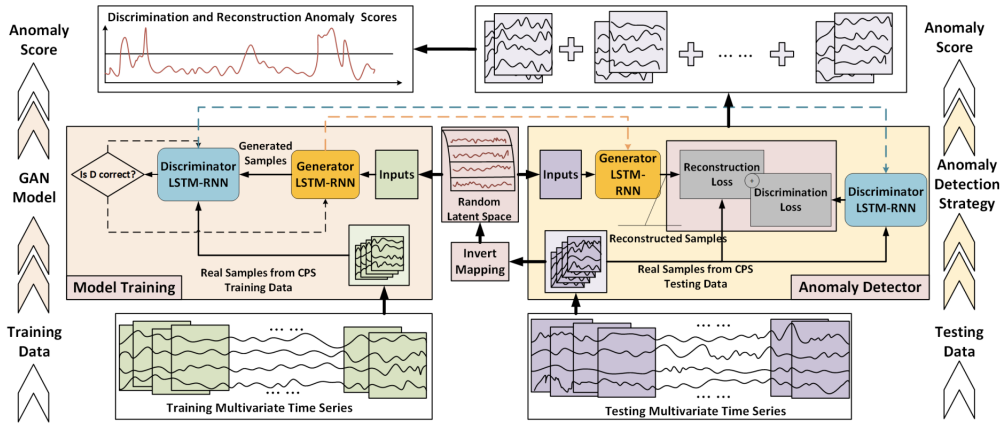


Figure 2.4: MAD-GAN: Unsupervised GAN-based anomaly detection architecture

In Figure 2.4 GAN framework is shown on the left, and the generator and discriminator were learned by iterative adversarial training. The process for detecting anomalies is shown on the right, where a combined anomaly score based on discrimination and reconstruction is computed using both the GAN-trained discriminator and the generator.

Discrimination and Reconstruction Anomaly Score (DR-Score), a new metric that MAD-GAN uses to identify anomalies, is introduced. This score integrates data from the discrimination and reconstruction procedures, helping to detect potential data anomalies. The DR-Score helps MAD-GAN outperform other approaches in finding abnormalities.

In conclusion, the MAD-GAN technique represents the potential to distinguish abnormalities in complex datasets. Data from networked sensors and actuators can be analyzed to find anomalous activity that might point to many internal/external system-related problems. Moreover, this approach provides a more precise and effective means of ensuring the dependability and safety of these critical systems.

2.8 Wav2Vec 2.0

Wav2Vec 2.0[6] is a CNN-based deep learning model that utilizes Transformer Encoder developed by Facebook for speech recognition tasks. It uses a self-supervised learning approach to learn representations from masked raw audio data. The model suggested by Baevski et al. [10], "Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations".

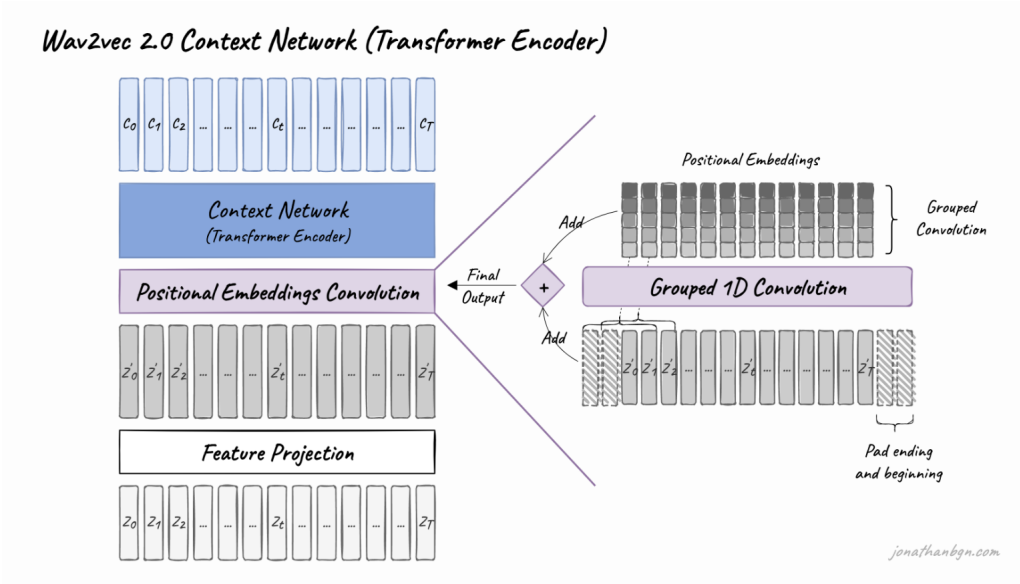


Figure 2.5: The Feature Encoder Architecture of Wav2Vec 2.0

Wav2Vec 2.0 model is suitable for univariate time series analysis. In our study, we intended to use the Wav2Vec 2.0 pre-trained model to investigate the latent representation quality of a univariate time series analysis model on multivariate time series data by using a sliding window approach to capture temporal dependencies and intricate features, right after following a fusion of features approach.

As shown in Figure 2.5, the Transformer Encoder layer uses positional embedding to capture the temporal dependencies. For pretrained versions, two different versions are publicly available, the 'BASE' and 'LARGE' models. During our experiments, the size of the datasets we used for all three public datasets was more suitable for using the 'BASE' pretrained version of the Wav2Vec model for feature embedding. Therefore, it is determined to conduct our experiments using the 'BASE' Wav2Vec model.

2.9 ResNet50

ResNet50[4] is one of the optimal CNN-based deep learning models while finding optimal solutions and eases optimization by providing faster convergence in the early stage [61]. It was first suggested by He et al.[26] to be used in image classification. We intended to use by utilizing the pre-trained 'ImageNet' model's weights to generate window-based embeddings (representations). During the generation of representations, we first converted time series windows into Heatmaps according to the window size specified in our experiments and before feeding them into the ResNet50 model. Then, by getting the output of the second-to-last layer of the model, also known as the penultimate layer, we generated window-based representations using the ResNet50 model. The output of this layer is often chosen because it contains rich feature representations learned from the data while being abstract enough to be applied to various tasks. [61]

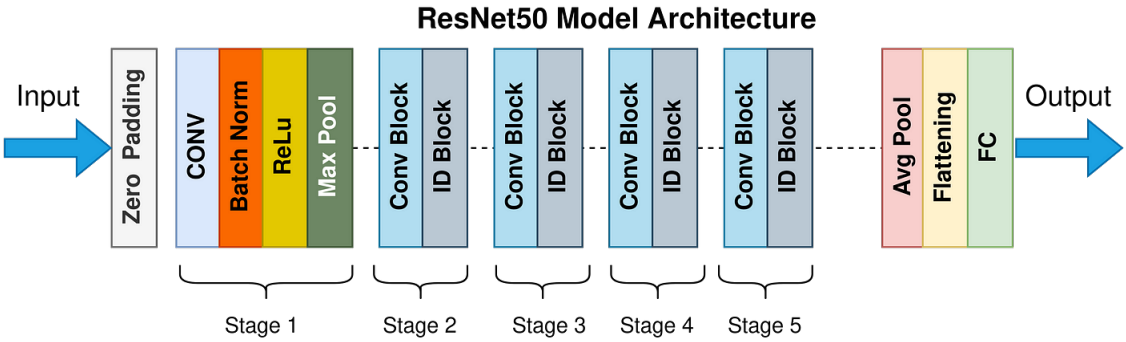


Figure 2.6: The Architecture of ResNet50 Model

Different types of layers can be preferred to decrease training/validation errors during the training procedure [61]. The most used layer types are ResNet34, ResNet50, ResNet101 and ResNet152. To pick the suitable ResNet model for our experiments we considered pre-trained models used in the literature for multivariate time series analysis. Chaudhuri et al. [8] showed in their study that for multivariate time series analysis, the ResNet50 model had a higher fault classification accuracy than the VGG16 model[48]. Furthermore, Boesch et al. [12] notes that the ResNet50 model introduced a three-layer bottleneck design which allows the ResNet50 model to have a higher classification accuracy than the ResNet34 model. Lastly, ResNet101 and ResNet152 are used in more complex datasets, and image classification problems.[12] Since the three public datasets used in our experiments were more suitable for less complex datasets compared to ResNet101 and ResNet152, we decided to use the ResNet50 model to generate representations.

Note that these methodologies are part of ongoing research. Their effectiveness may vary depending on the specific characteristics of the time series data and the complexity of the problem at hand.

Chapter 3

Methodology

In previous chapters, we introduced the problem description, our research questions, and the research approach. Later, we established the background for this research. In this chapter, we present the methodology that guides our research.

In our study, we outline the methodology for measuring the impact of windowing on classification performance and representation qualities of deep learning models in three phases. First, we start with a description of the three datasets we used in our experiments, followed by a detailed explanation of the dataset preparation process prior to the classification phase. Here, we motivate our selection and the necessary preprocessing steps applied to those datasets. Second, we explain how we retrain and test SOTA classification models and utilize pre-trained models to investigate the impact of window size on classification results, and evaluate them based on the evaluation metrics we explained in Chapter 2, Third, after collecting representations generated during the model testing process, we present novel quantitative metrics to measure representation qualities and how to visually support those metrics with dimensionality reduction plots.

3.1 Dataset Descriptions

To rigorously assess the impact of windowing on multivariate time series fault detection, we selected three open source multivariate time series anomaly datasets: SMD, SMAP, and MSL [52, 28]. These datasets cover real-world machine data and expert-labeled telemetry satellite data from NASA, offer a diverse and comprehensive range of data. The SMD dataset, sourced from an Internet company, provides a realistic example of server machine data. The SMAP and MSL datasets, both from NASA, add credibility to our research and introduce the challenges of satellite telemetry data. The varied anomaly ratios across these datasets, combined with their inherent characteristics of real-life anomaly data, ensure a robust evaluation of the models and allow for generic research findings. The following datasets are used in our experiments:

Dataset Name	Train/Feature Size	Test/Feature Size	Test Label Size	The Machine Used	Anomaly Ratio
Soil Moisture Active Passive Satellite(SMAP)[52]	(2880, 25)	(8640, 25)	8640	E-13	3.03%
Mars Science Laboratory Rover (MSL)[52]	(4308, 25)	(6100, 25)	6100	P-10	2.14%
Server Machine (SMD)[28]	(28479, 25)	(28479, 25)	28479	machine-1-1	9.36%

Table 3.1: The Information of Three Public Datasets Used in Our Experiments

The three public datasets provided a rich source of samples for testing pre-trained classification models, as well as for training and evaluating the performance of state-of-the-art multivariate time series classification models and latent representation generation processes. By using these three datasets, we included a variety of sizes and anomaly rates in our research. In particular, in Experiment Representation Quality Evaluation, we observed that the diversity of the datasets was reflected in the classification results and the latent representation qualities.

During the preprocessing stage, we faced a decision regarding the inclusion of all machines and entities from the datasets. For our experimental setup, prioritizing generalizability and cost-effectiveness (memory usage), we selected a machine with the most training and test data from each dataset. This approach provided a consistent basis for comparison in all experiments and ensured that every chosen machine is representative of the corresponding dataset. Note that in Table 3.1, we share the sizes of the datasets we used in our experiments. The sizes of open source datasets, including all machines and entities, are detailed in sections: 3.1.1, 3.1.2, 3.1.3

3.1.1 Server Machine Dataset (SMD)

SMD dataset is a 5-week-long machine data that was collected from an Internet company, and used in the OmniAnomaly paper, that we involved in our experiments[28]. In the extended SMD dataset, there are 708,420 test data points and 708,405 train data points with an anomaly ratio of 4.16%; and 95.84% normal data. There are 28 different entities (machines) with a feature size of 38 in the dataset. Following the OmniAnomaly paper's implementation[28], we used only the first machine's ('machine-1-1') training, testing, and test label datasets.

3.1.2 Soil Moisture Active Passive Satellite (SMAP)

SMAP dataset is an expert-labeled telemetry satellite anomaly data from NASA. The data collection frequency for the SMAP dataset is per 2-3 days. In the extended SMAP dataset, there are 427,617 test data points, and 135,183 train data points with an anomaly ratio of 13.13%; and 86.87% normal data. Additionally, the SMAP dataset has several 55 entities with a feature size of 25.[52] Training set does not include labels; therefore, it is intended to be used for training and unsupervised learning purposes. For our experiments, we decided to use only one machine namely 'E-13'. While selecting, we decided to pick a machine with the most training and test data for our experimental setup, generalizability, and real-life data representability.

3.1.3 Mars Science Laboratory Rover (MSL)

MSL dataset is another expert-labeled telemetry satellite anomaly data from NASA. In the extended MSL dataset, there are 58,317 training data points and 73,729 test data points with an anomaly ratio of 10.72% and 89.28% normal data. Additionally, the MSL dataset includes 3 different machines having a total number of 27 entities with a feature size of 33.[52] Training set does not include labels; therefore, it is intended for training and unsupervised learning purposes. During our experiments, we used machine data from the entity: 'P-10'. While selecting, we decided to pick a machine with the most training and test data for our experiment setup and real-life data representability.

3.2 Preprocessing Datasets

The three selected public datasets have unique dimensions and anomaly ratios. They originate from various domains, such as SMD data is server machine sensor data with a feature size of 38, and MSL and SMAP are public datasets from NASA, with dimensions 33 and 25, respectively. In our study, we aim our analysis to be robust and directly align with our research questions starting with the preprocessing phase. We ensured that the insights we drew were truly a result of our experiments, particularly in terms of window size, and not from any inconsistencies in the

datasets. To accurately assess the impact of window size on classification performance and latent representation qualities, we focused on the most important features, eliminating potential noise or redundancies, to base our conclusions on the most informative aspects of the data. Lastly, due to the computational demands of the ResNet50 model (see Section 3.2.2), we opted for strategic sampling of the datasets to maintain the integrity of our analysis. Therefore, before starting our experiments, we applied several preprocessing techniques to our input datasets such as feature selection for datasets having different feature sizes, normalization, and sampling explained in detail in Sections 3.2.1, and 3.2.2.

3.2.1 Feature Selection for MSL and SMD Datasets

During the data exploration phase of our experiments, we observed that the MSL and SMD datasets contained feature columns with only zero values. Accordingly, we decided to eliminate these empty columns and reduced the number of features for the SMD (originally 38 features) and MSL (originally 55 features) datasets. Our goal was to align the feature column sizes of all datasets with the SMAP dataset, which had 25 features. To achieve this, we selected the top 25 features based on two criteria: the lowest correlation and the highest standard deviation. The rationale for this decision was to ensure consistency in all our experiments [40]. By maintaining equal-sized features for all datasets, each window of data processed by our models could have the same dimension. This consistency allowed us to compare every window with the same size and ensured that our experiments with classification models were uniform. In addition to this, our motivation was not only to achieve uniformity in size, but also to create a controlled experimental environment [15]. By standardizing the size of the window, we can more accurately measure the impact of windowing on our results [29]. Although we recognize that the inherent data complexity might differ across datasets, having a consistent window size ensures that any observed effects in our results are attributed to the model's performance and not affected by the variability in data window sizes.

3.2.2 Sampling for ResNet50 Representation Generation

ResNet50 model is built to classify images for large and complex systems involving convolutional neural networks (CNNs). To use the ResNet50 model, we created heatmaps of the windows and resized them to (224, 224), the input size of the pre-trained ResNet50 model. Considering the extended period of time of representation generation (it takes weeks) process due to the time it takes for the compulsory process of converting the time series windows into heatmaps, saving and re-reading a created heatmap, it is decided to sample three public datasets to lower the representation generation time.

Figure 3.1: An Example Heatmap of A Window (Sized 4) Fed into Pre-trained ResNet50 Model to Generate A Representation

In order to avoid loss of generality, we utilized the StratifiedKFold method of the sklearn library[5] to split all datasets into 10 equal-sized folds and randomly pick one fold that has the preserved distribution of anomaly data with the original datasets. Therefore, we preserved the original test data anomaly distribution and sampled the dataset with a size of 0.1 proportions. Particularly for the SMD dataset, we took 0.04 percent, considering the test size of 28479. Therefore, we used a stratified sample of size 1140. For the SMAP dataset, the test data size was sampled from 8640 to 864; and for the MSL dataset from 6100 to 610. As we sampled three data sets while preserving the original test data distributions, we did not observe any performance differences after sampling.

3.2.3 Sliding (Overlapping) Window Approach

Upon completion of the aforementioned preprocessing steps, as VAE, GAN, and CNN have complex structures, and the common practice is to feed time series data into deep learning models as batches of windows during training and testing processes [7]. We adopted a sliding window approach, a technique that segments the time series data into overlapping windows, advancing one time step at a time (in our study). This method is particularly beneficial as it provides a more granular view of the data and ensures that subtle patterns within the data are not overlooked [7]. Especially when training models that include a Recurrent Neural Network layer, such as the GRU (Gated Recurrent Unit) for OmniAnomaly, and GANs for the MAD-GAN model, the data needs to be reshaped. Thus, we flattened the windows to achieve an input size of (1, feature-size * window-size) before feeding them into the models. In summary, using a sliding window approach allowed us to quantify and measure the impact of the window size while evaluating the classification performances and latent representation qualities.

3.3 Schematic Overview of Train and Test Protocols

In this section, we present a schematic overview of the training and testing protocols for all deep learning models used to complete our experiments.

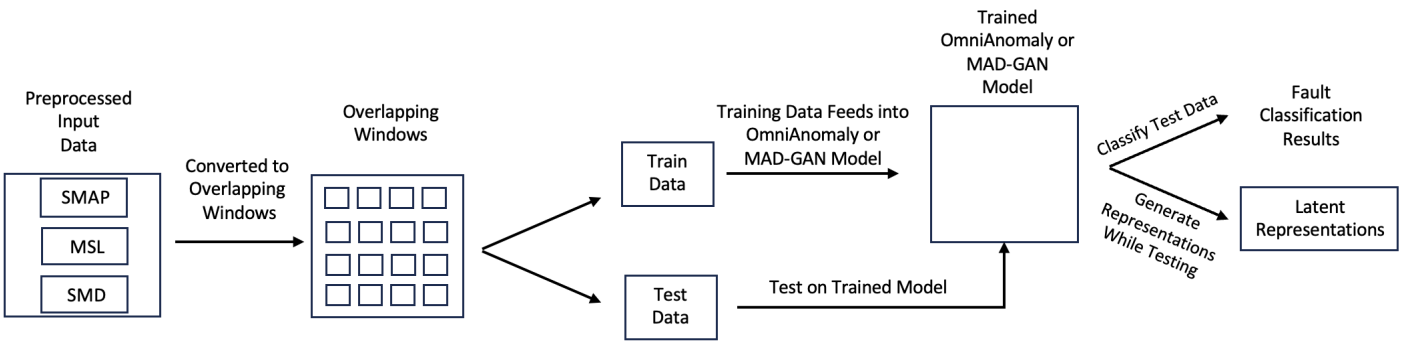


Figure 3.2: Schematic Overview of Train and Test Protocols For SOTA Models

In Experiment SOTA Classification, we used SOTA fault classification models, namely OmniAnomaly and MAD-GAN, to evaluate their fault classification performances. As shown in Figure 3.2, after following the preprocessing steps that we explain in Section 3.2, the preprocessed data are converted into overlapping windows (both train and test data) with a distinct window size. Later, the train data is used to train OmniAnomaly/MAD-GAN model, and test data is used to generate latent representations and fault classification results. This training and testing routine was repeated three times (with different seed values: 1, 1000, 100000 to initialize weights) for each distinct window size experiment.

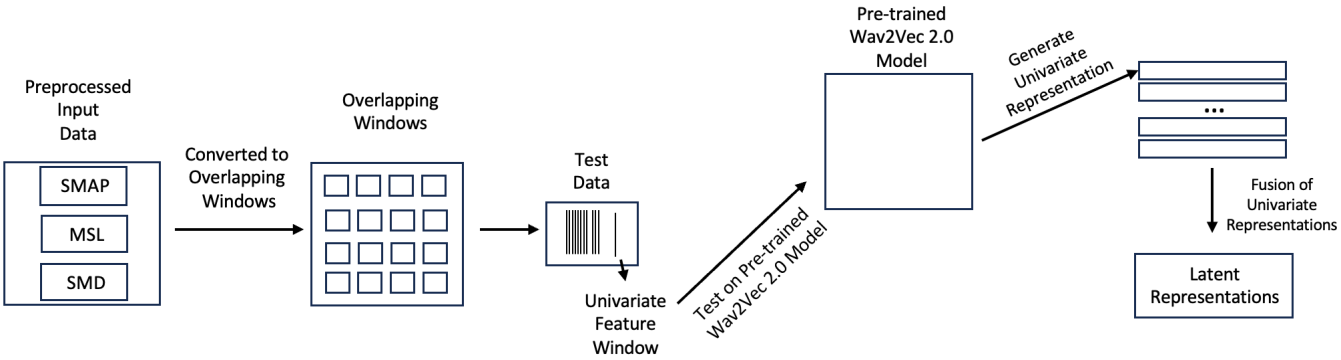


Figure 3.3: Schematic Overview of Train and Test Protocols For Wav2Vec 2.0 Model

In Experiment Pre-trained Classification, we used pre-trained Wav2Vec 2.0 and ResNet50 deep learning models to evaluate the windowing impact on their fault classification performances. For the Wav2Vec 2.0 model, as shown in Figure 3.3, after preprocessing and converting the data into overlapping windows, we took 1 of the 25 feature columns within one window and fed into the model. The motivation behind this is that the Wav2Vec 2.0 model is a pre-trained speech recognition model developed for univariate time series data. To generate representations for each window, we followed the fusion of 25 univariate feature representations approach and applied this to all other windows.

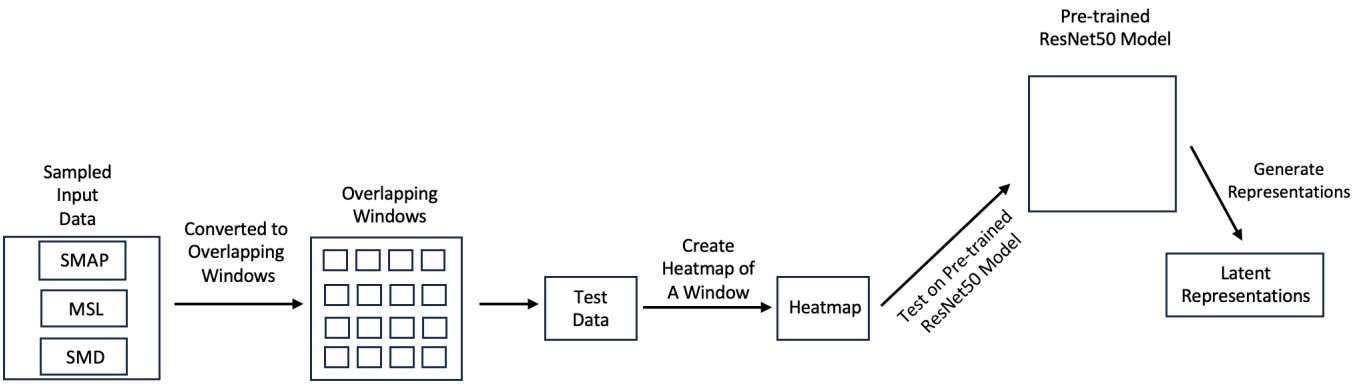


Figure 3.4: Schematic Overview of Train and Test Protocols For ResNet50 Model

As shown in Figure 3.4, for the ResNet50 model, we sampled the dataset ensuring no loss of generality as detailed in Section 3.2.2. After converting the data into overlapping windows, we created Heatmaps for each window and fed them into the model. Note that we explain how to evaluate the classification performances of pre-trained models utilizing Prototypical Networks approach in Section 3.4

After collecting representations that are generated after the steps shown in Figures: 3.2, 3.3, 3.4, we continue with the Experiment Representation Quality Evaluation, where we investigate the impact of window size on representation qualities by presenting quantitative metrics that are explained in Section 3.5.

3.4 Prototypical Networks

In our study, we utilized the meta-knowledge from multiple auxiliary networks suggested by Ding et al. [20], who successfully applied Prototypical Networks for few-shot network anomaly detection, and adapted the learning from their study to our research context. Similarly, we lacked labels for the representation clusters generated by the pre-trained models, namely Wav2Vec 2.0[6] and ResNet50[4]. Therefore, we used Prototypical Networks to create labels for time series representations generated by pre-trained models.

Prototypical Networks, introduced by Snell et al.[49], are a type of few-shot learning model that leverages the concept of a prototype, or a representative example of a class, for classification tasks. In our study, we employed the classification approach used in the Prototypical Networks, a type of few-shot learning model [49], to calculate the log-likelihood of distances from a point in window-based representations. We utilized log-likelihood distances to classify each point as either an anomaly (label 1) or a non-anomaly (label 0) based on their distances to centroids of each class.

3.5 Centroid-Based Representation Quality Metrics

Existing methods such as statistical tests, evaluating classification results using latent representations, and visualization using dimensionality reduction techniques can be used for quality measures. We first considered dimensionality reduction methods like t-SNE and UMAP as quality measures for representations.

3.5.1 Picking t-SNE For Representation Visualization

We employed the t-SNE method to visually interpret the representation qualities. The purpose behind picking the t-SNE dimensionality reduction method is its ability to capture local patterns and maintain clusters. Wang et al. [56] suggested in their study, unlike UMAP, t-SNE is one of the local dimensionality reduction methods which can highlight densely packed regions or outliers better. Furthermore, t-SNE is more effective on smaller datasets compared to UMAP. [56]Therefore, we selected t-SNE as the dimensionality reduction method to plot latent representations to enhance the understandability of latent representations.

Although dimensionality reduction techniques successfully visualize high-dimensional data in 2D and allows better interpretability for representations, we observed they need supportive evidence such as quantitative measures for fault detection.

The classification results allow us to calculate centroids for each class (cluster). Therefore, based on the centroids for each class, we introduced two centroid-based measurement metrics to evaluate representation qualities:

1. The Minimum Inter-Centroid Distance
2. The Maximum Standard Deviation of Intra-Centroid Distances

The assumptions behind those metrics are:

- Each class for good representations should be highly separable from others, meaning that centroids distances (inter-centroid distances) should be large enough.
- Samples within the class for good representations should be close to each other, that is, sample-to-centroids (intra-centroid-distance) distances should be small enough.

Our motivation behind picking those metrics is to consider a worst-case evaluation metric as a more reasonable evaluation criterion. Using maximum and minimum values offers insight into data extremities, vital for worst-case scenarios. Maxima, in the context of intra-centroid distances,

highlight the broadest spread within clusters, ensuring even the most dispersed cluster remains compact. Minima, for inter-centroid distances, emphasize the nearest distance between clusters, guaranteeing distinct separation. Averages, on the other hand, might obscure critical outliers, giving a generalized view and possibly missing poorly represented clusters [43].

To elaborate on metrics, we expected quality representations should have smaller variations, thus small standard deviations of distances between a centroid representing the associated cluster and the representations' data points within clusters. Additionally, we expected classes to be highly-separable (having higher distances), which indicates quality representations.

Regarding the calculations for "The Maximum Standard Deviation of Intra-Centroid Distances" metric, we first found centroids by considering Euclidean Distances. Later, we calculated the distances within each cluster and took the standard deviation of distances for the respective clusters. Finally, we collected the maximum value of standard deviations of distances for both the anomaly(1) and non-anomaly(0) clusters and picked the higher value. For "The Minimum Inter-Centroid Distance" metric, we similarly found centroids for the anomaly(1) and non-anomaly(0) clusters and calculated their Euclidean Distances.

Our intention in choosing both metrics was to compare the qualities of different representations generated by four deep-learning models. To validate our intuition, we also used the t-SNE dimensionality reduction method to visualize window-based latent representations. We share the results and our findings from centroid-based metrics in Section 4.3.

Chapter 4

Results and Discussion

In this chapter, we present our three experiments:

- In Experiment SOTA Classification, we investigate the relationships between windowing impact and multivariate time series classification models.
- In Experiment Pre-trained Classification, using the Prototypical Networks approach [49], we first generated labels. Later, using generated labels, we explore the relationships between window size and classification performances for pre-trained models.
- Regarding the experimental setup in Experiment SOTA Classification and Experiment Pre-trained Classification, we ran our experiments three times to get consistent results for different random seed values (1, 1000, and 100000).
- During Experiment SOTA Classification and Experiment Pre-trained Classification, we collected latent representations and used representations to evaluate their qualities in Experiment Representation Quality Evaluation.
- In Experiment Representation Quality Evaluation, we evaluate the quality of latent representations generated by four models with distinct window sizes and investigate the representation qualities using our quantitative metrics. Then we utilized t-SNE to plot 2D visualizations and visually validate quantitative metrics.

Each section starts with a presentation of experimental results in tabular form. Following that, we extract insights from these tables and articulate our observations. To further elucidate our findings, we also provide line plots derived from result tables and figures created during our experiments for a detailed discussion of the findings of each experiment. Finally, at the end of the discussion section of Experiment Representation Quality Evaluation, we discuss the results by comparing quantitative and qualitative evaluations of latent representation qualities.

4.1 Experiment SOTA Classification

In this section, we present the results from Experiment SOTA Classification. In the next section, we provide the classification results in a table. Here, the classification results were evaluated using G-Mean, Recall, and Specificity metrics across five different window sizes. A higher G-Mean score indicates better classification performance. While a higher Recall value indicates the model can detect anomalies better, a higher Specificity value indicates the model can accurately identify normal instances. Later on, we share our observations from the results. In the discussion section, we provide a line plot of G-Mean scores for the classification models. In these plots, the mean of three G-Mean scores derived from each experiment for every window size is shown in horizontal

lines. In addition, we plotted the standard deviation of three G-Mean scores in vertical lines. Finally, we discuss our findings from the results of Experiment SOTA Classification.

In Experiment SOTA Classification, we defined our study to find an answer to the first research question:

RQ1) How does the window size parameter impact current state-of-the-art multivariate time series classification models?

In Experiment SOTA Classification, our primary focus is to track the impact of windowing on model classification performances for SOTA multivariate time series classification models.

4.1.1 Results For Experiment SOTA Classification

In the results section, we start by sharing the results of Experiment SOTA Classification from the OmniAnomaly model, and the MAD-GAN model in tabular formats. Later, we respectively share our observations and discuss our findings from Experiment SOTA Classification in the discussion section.

Before sharing the results for Experiment SOTA Classification, we would like to address the reason behind the Recall results shown in Tables 4.1 and 4.2 are quite high (e.g. 0,9990) and remain consistent across different window sizes. There are several reasons for this.

First, we employ SOTA fault classification models, which excel in detecting anomalies in multivariate time series data. This is consistent with the findings of Tuli et al. [54], who also examined several SOTA classification models. Using Recall as their evaluation metric, Tuli et al. found that the performance of the OmniAnomaly and MAD-GAN models on the SMD, SMAP, and MSL datasets aligns closely with our results.

Second, with respect to consistent recall results across different window sizes, during our training and testing process, we used 4 decimals after "0,..." and performed the evaluation metric calculations. Accordingly, the Recall results we received in our experiments are approximate values ($\sim$ 0,....) (see Figures A.1 and A.2 in the Appendix).

Third, in our experiments, we chose to use the G-Mean metric due to its significance in addressing imbalanced datasets, a prevalent challenge in multivariate time series fault classification. Especially in anomaly detection, anomalies (positive class) are often much fewer than normal data points (negative class). This imbalance can cause models to be biased towards the majority class, yielding high accuracy but fails to detect anomalies. The G-Mean metric helps balance performance between both classes by emphasizing recall and specificity. A high G-Mean indicates that a model effectively identifies both anomalies and normal data points. Our results showed that the SOTA model successfully detected most of the actual anomalies, thus reducing false negatives, which occur when an anomaly is incorrectly classified as a normal data point. In addition to this, it minimized false positives, which arise when a normal data point is wrongly labeled as an anomaly. Given this balanced performance, while precision remains an informative metric, our primary focus on recall and specificity offers the essential insights required for imbalanced datasets.

4.1.2 Results For OmniAnomaly

Considering Table 4.1, after training the OmniAnomaly model, overall, test results show consistent performances for OmniAnomaly models across distinct window sizes, as shown in Table 4.1. As an exception, when the OmniAnomaly model was trained with the SMAP data, which have the smallest training data in our experiments, for the window size 4, the test results for the G-Mean score and the Specificity score displayed a higher standard deviation (0,0013 and 0,0022). On the other hand, the OmniAnomaly model had decreasing G-Mean scores when trained and tested with the MSL dataset for windows from 4 to 600.

As shown in Table 4.1, we see consistent results for OmniAnomaly models in the three datasets for the SOTA Classification Experiment.

The observations for the OmniAnomaly,

- Our experiments showed the OmniAnomaly model has more robust results (smaller standard deviation of results) compared to the MAD-GAN model for distinct window sizes.
- For the SMD dataset, which has the most training and testing data, classification performance increased when the window size also increased considering the G-Mean metric.
- SMAP dataset has the smallest training size. It is observed that the model was more sensitive to changes in window size when window size was set to 4 compared to MSL and SMD datasets. Until the window size of 100, the G-Mean score increased and slightly decreased at window size 400 and increased again at 600.
- Although the window size was increased during Experiment SOTA Classification, recall values for all three datasets remained the same and displayed consistency.
- Considering the training size with having the largest number of training and testing windows for the SMD dataset, the OmniAnomaly model has the highest G-mean score for detecting anomalies among the three datasets. Also, for all three performance evaluation metrics, excluding the Specificity score.

Dataset	Score	Result Type	Window_Size_4	Window_Size_40	Window_Size_100	Window_Size_400	Window_Size_600
MSL	G-Mean	Mean	0,8725	0,8727	0,8723	0,8650	0,8596
		Std	0,0013	0,0002	0,0002	0,0000	0,0001
	Recall	Mean	0,9999	0,9999	0,9999	0,9999	0,9999
		Std	0,0000	0,0000	0,0000	0,0000	0,0000
	Specificity	Mean	0,7613	0,7616	0,7611	0,7483	0,7392
		Std	0,0022	0,0005	0,0001	0,0001	0,0004
SMAP	G-Mean	Mean	0,9097	0,9164	0,9160	0,9170	0,9164
		Std	0,0114	0,0009	0,0007	0,0006	0,0002
	Recall	Mean	0,8467	0,8467	0,8467	0,8467	0,8467
		Std	0,0000	0,0000	0,0000	0,0000	0,0000
	Specificity	Mean	0,9776	0,9919	0,9911	0,9932	0,9917
		Std	0,0243	0,0019	0,0015	0,0014	0,0005
SMD	G-Mean	Mean	0,9819	0,9879	0,9915	0,9924	0,9921
		Std	0,0034	0,0026	0,0010	0,0020	0,0001
	Recall	Mean	0,9999	0,9999	0,9999	0,9999	0,9999
		Std	0,0000	0,0000	0,0000	0,0000	0,0000
	Specificity	Mean	0,9644	0,9759	0,9847	0,9849	0,9843
		Std	0.0065	0,0051	0,0021	0,0040	0,0001

Table 4.1: OmniAnomaly Experiment SOTA Classification Results

4.1.3 Results For MAD-GAN

As shown in Table 4.2, we observed fluctuating results observed for different datasets during Experiment SOTA Classification for the MAD-GAN model.

Dataset	Score	Result Type	Window_Size_4	Window_Size_40	Window_Size_100	Window_Size_400	Window_Size_600
MSL	G-Mean	Mean	0,8660	0,8524	0,8483	0,8224	0,7702
		Std	0,0001	0,0007	0,0077	0,0206	0,0054
	Recall	Mean	0,9999	0,9999	0,9999	0,9993	0,9999
		Std	0,0000	0,0000	0,0000	0,0005	0,0000
	Specificity	Mean	0,7500	0,7266	0,7197	0,6766	0,5933
		Std	0,0003	0,0011	0,0130	0,0341	0,0084
SMAP	G-Mean	Mean	0,9390	0,9853	0,9818	0,8716	0,9415
		Std	0,0460	0,0019	0,0011	0,0050	0,0098
	Recall	Mean	0,8978	0,9999	0,9999	0,8467	0,9999
		Std	0,0885	0,0000	0,0000	0,0000	0,0000
	Specificity	Mean	0,9837	0,9709	0,9639	0,8973	0,8861
		Std	0,0007	0,0037	0,0021	0,0104	0,0190
SMD	G-Mean	Mean	0,9923	0,9981	0,9690	0,8807	0,8470
		Std	0,0037	0,0003	0,0439	0,0655	0,0607
	Recall	Mean	0,9973	0,9973	0,9462	0,7791	0,7297
		Std	0,0000	0,0000	0,0885	0,1124	0,0990
	Specificity	Mean	0,9872	0,9990	0,9953	0,9991	0,9856
		Std	0,0074	0,0005	0.0100	0,0011	0,0157

Table 4.2: MAD-GAN Experiment SOTA Classification Results

The observations for the MAD-GAN,

- MAD-GAN model was more sensitive to window size alterations compared to the OmniAnomaly model.
- When the window size increased to 40 and 100, the MAD-GAN model performed the best across three metrics for the SMAP dataset, excluding the Specificity score. However, there was a considerable decrease in the G mean scores when we increased the window size to 400, then a slight increase at 600.
- We observed the highest fluctuations in the results for the MAD-GAN model trained on the SMD dataset, which has the most training and testing data. According to our results, for window size 100 (also for 400 and 600), the model performed various scores with high standard deviations. Yet, the MAD-GAN model performed consistently according to the Specificity score metric. In other words, the MAD-GAN model identified normal data points better than identifying anomaly data points for window sizes 400 and 600.
- For the MSL dataset, when the window size increased, the fault classification performance of the MAD-GAN model decreased, similar to the OmniAnomaly model.

Consequently, Experiment SOTA Classification gave us significant observations about window size and the windowing impact on classification performances. Those observations will be discussed in detail in the discussions.

4.1.4 Discussion For Experiment SOTA Classification

Regarding the effects of windowing on multivariate time series classification models, Experiment SOTA Classification produced significant conclusions such as OmniAnomaly stands out as a robust fault classification model. Furthermore, we extracted information about the impact of diversity in training datasets on classification performances. The classification results revealed the relationship

between the window size and the fault classification performance of the SOTA models. Therefore, we were able to find answers to the first research question.

Results for the OmniAnomaly model initially displayed in Figure 4.1, we observed high G-mean scores and low standard deviation along different window sizes, which presents an indication for consistent classification performance. However, we observed a higher standard deviation, with an exception when the window size was set to 4, compared to other G-Mean scores derived from three datasets using the OmniAnomaly model.

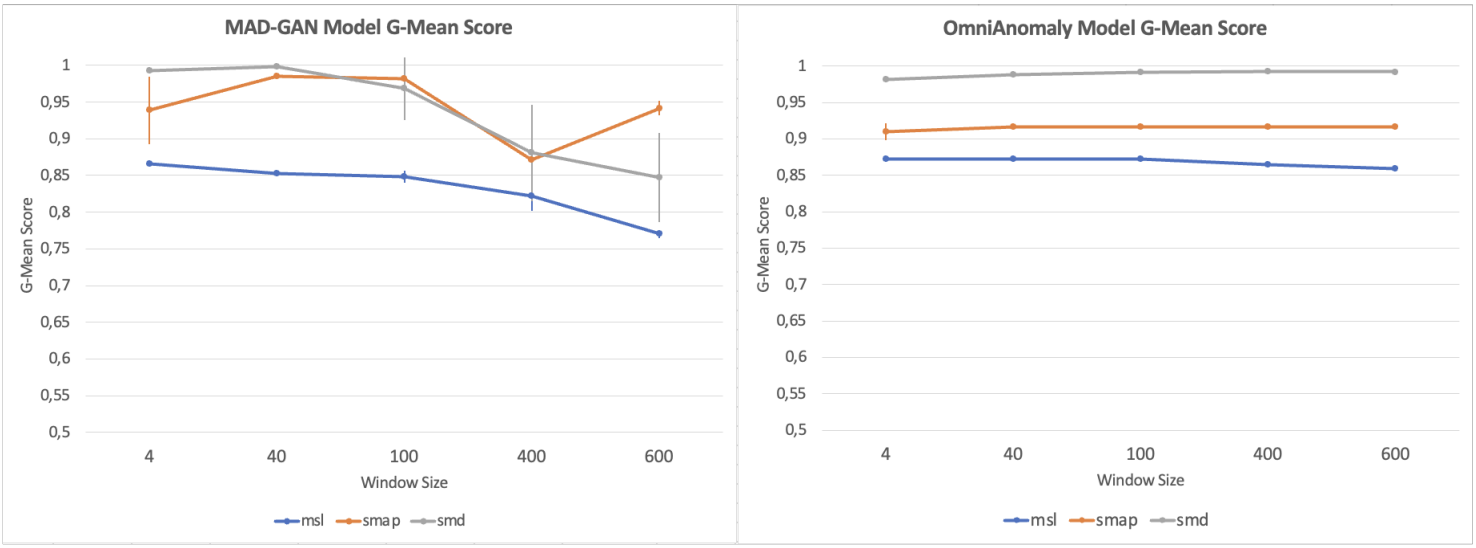
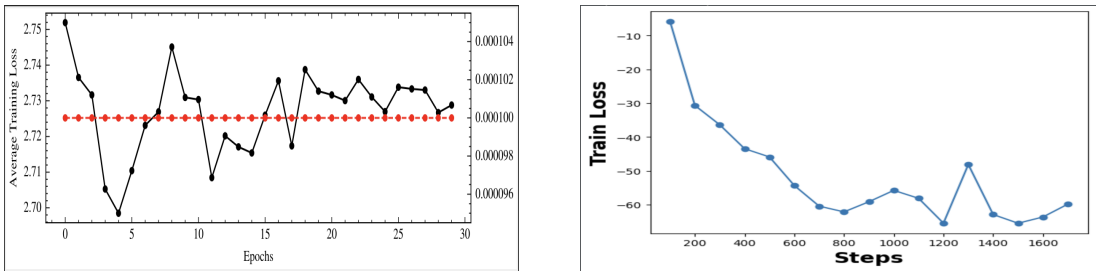


Figure 4.1: OmniAnomaly and MAD-GAN Models' Experiment SOTA Classification Results

On the other hand, the MAD-GAN model's classification results with different window sizes were inconclusive due to the mode collapse we observed during the training process in our experiments. As shown in Figure 4.1, the MAD-GAN model displayed inconsistent results with high variations which indicates the MAD-GAN model is an unstable model for fault classification according to the results of Experiment SOTA Classification. Moreover, during our experiments, we ran the MAD-GAN model for 30 epochs to observe the mode collapse with a high effort for hyper-parameter tuning which makes the MAD-GAN a hard-to-train model.



(a) MAD-GAN Model

(b) OmniAnomaly Model

Figure 4.2: Training Loss Curve Examples For SOTA Classification Models

The OmniAnomaly model is a consistent model, that converged during our experiments. We provided Figure 4.2 (b) to show the training loss curve of the OmniAnomaly model. As shown in the training curve, the training loss gradually decreases in every step and saturates. Conversely, in the MAD-GAN training loss curve in Figure 4.2 (a), we observed a decrease in the training loss

until a point while the model is training in epochs. Herein, after the fifth epoch, we observed a common failure of GAN models "mode collapse", where the generator network fails to produce diverse and varied samples.[39] Thus, the model starts to produce fluctuating training loss values and shows an unstable model classification performance.

In order to solve mode collapse in GANs, fine-tuning the GAN model's hyper-parameters is a crucial approach. By appropriately picking the learning rate (lr) hyper-parameter, a GAN model can be fine-tuned and might overcome the mode collapse problem, thus promoting a more diversified sample generation. However, this practice leads to high training costs, which also makes the MAD-GAN model a challenging model to train and an unstable model.

4.2 Experiment Pre-trained Classification

The primary objective of Experiment Pre-trained Classification is to track the impact of windowing on model classification performances for the pre-trained models utilizing Prototypical Networks. Pre-trained models' classification performances are crucial in shedding light on their future usability in cases such as insufficient resources or research seeking a cost-effective multivariate time series analysis method.

In Experiment Pre-trained Classification, we defined our study to find an answer to the second research question:

RQ2) How does the window size parameter impact pre-trained multivariate time series models' classification performance?

Before examining the results of Experiment Pre-trained Classification and finding an answer to the second research question, we first would like to address several input specifications for pre-trained models, such as testing the windowing impact of latent representation qualities and the representation generation steps.

Wav2Vec 2.0[6] is a state-of-the-art audio classification model developed by the Facebook team. Its pre-trained version is available on the Transformers package of hugging-face[2]. The pre-trained model had the following input specifications:

- The input data must have a minimum window size of 400 to be eligible to use the pre-trained 'BASE' version of the Wav2Vec 2.0 model.
- Wav2Vec 2.0 model developed for univariate time series data, it is required to use a fusion of features approach while creating representations for multivariate time series data.
- Generated outputs occupy large storage spaces, which are inefficient for machines with limited storage capacities. (Wav2Vec 2.0 representations for SMD dataset for the window size 600 occupies 2.01 GB storage space)

Additionally, the ResNet50 model[4] is a popular CNN-based model that is used for image classification. It can be used for time series classification and representation generation after applying the pre-processing stages explained in Chapter 3. For the input data, to generate representations, it is required to first convert time series data into windows, then create heatmaps of windows, and lastly resize the input image to (224, 224) to generate the representation of a multivariate time series data.

Similarly, the ResNet50 model also has several input specifications:

- The process of converting the window into an image and reading from the local machine takes time. It can take days for the training process for larger window sizes, such as 600 and 400. Also, it takes approximately a week to generate representations for window size 4 without sampling.

- To tackle the time constraint, our 3-time series datasets were sampled utilizing StratifiedKFold[5] method from sklearn without losing the generalizability. Thus, the distribution of the original anomaly ratios in the original datasets is preserved.

After considering the input specifications of the pre-trained models, in the following sections, we provide the classification results in a table. Here, the classification results were evaluated using G-Mean, Recall, and Specificity metrics across five different window sizes. A higher G-Mean score indicates better classification performance. While a higher Recall value indicates the model can detect anomalies better, a higher Specificity value indicates the model can accurately identify normal instances. We first share our observations from the results of Experiment Pre-trained Classification. Later in the discussion section, we provide a line plot of G-Mean scores for the pre-trained models. In these plots, we compared the windowing impact on G-Mean scores for Wav2Vec 2.0 and ResNet50 models. Finally, we discuss our findings from the results of Experiment Pre-trained Classification.

4.2.1 Results For Experiment Pre-trained Classification

In Experiment Pre-trained Classification, as explained in Chapter 3, we utilized the Prototypical Network approach to generate anomaly labels for the representations generated by pre-trained models. This approach provided insights into representation classification practice and allowed us to compare the classification performances of the pre-trained models with the SOTA multivariate time series classification model's results from Experiment SOTA Classification.

As shown in Table 4.3, pre-trained models' (Wav2Vec 2.0 and ResNet50) classification results on three public datasets across three evaluation metrics are shared. We observed for the ResNet50 model when the window size was set to 600, the G-Mean scores resulted in 0. The reason behind this is if the size of representation samples is smaller, due to the Prototypical Network approach, all samples create a single cluster (anomaly (1) or non-anomaly (0)). To see the representation sample sizes and their class distributions, see Figure 4.4.

Dataset	Score	Pre-trained Models	Window_Size_40	Window_Size_40	Window_Size_100	Window_Size_400	Window_Size_600
MSL	G-Mean	Wav2Vec	-	-	-	0,9611	0,9489
		ResNet50	0,7417	0,8434	0,9633	0,9925	0
	Recall	Wav2Vec	-	-	-	0,9357	0,9067
		ResNet50	0,6875	0,9615	0,9821	0,985	0
	Specificity	Wav2Vec	-	-	-	0,9872	0,993
		ResNet50	0,8003	0,7398	0,9448	1	1
SMAP	G-Mean	Wav2Vec	-	-	-	0,6934	0,7139
		ResNet50	0,5679	0,6336	0,5078	0,5316	0
	Recall	Wav2Vec	-	-	-	0,7622	0,7228
		ResNet50	0,5142	0,768	0,927	0,2826	0
	Specificity	Wav2Vec	-	-	-	0,6308	0,7051
		ResNet50	0,6271	0,5228	0,2781	1	1
SMD	G-Mean	Wav2Vec	-	-	-	0,8045	0,7658
		ResNet50	0,8168	0,8828	0,8977	0,9368	0,9112
	Recall	Wav2Vec	-	-	-	0,7904	0,7344
		ResNet50	0,7704	0,8897	0,8694	0,8777	0,8303
	Specificity	Wav2Vec	-	-	-	0,8188	0,7987
		ResNet50	0,866	0,876	0,927	1	1

Table 4.3: Experiment Pre-trained Classification Results For Pre-trained Models

The observations for pre-trained models,

- The requirements for input specifications of the Wav2Vec 2.0 model for window sizes 4, 40, and 100 can be seen from line plots of G-Mean scores.
- ResNet50 model's classification results outperform Wav2Vec 2.0 model's classification results across all window sizes except window sizes 400 and 600 for the SMAP dataset. For the SMD

dataset, where results are more consistent compared to the other two datasets, the ResNet50 model's distinct outperformance can be seen.

- Wav2Vec 2.0 model's performance increases when window size increases from 400 to 600 except the SMAP dataset.

  Considering the ResNet50 model's performance,

- For the MSL dataset, G-Mean Score increases when the window size also increases to 400 and drops to 0 at 600.
- Also, for the SMD dataset, we observed an increase in the G-mean score when the window size increased to 400 and a slightly lowered score when it increased to 600.
- On the other hand, for the SMAP dataset, the G-mean score increases when the window size increases to 40. Then, we observed a decrease when we increased to 100 and increased to 400 and a sharp drop to 0 on 600. We also observed that Wav2Vec 2.0 model outperformed the ResNet50 model. The results for the SMAP dataset were observed to be inconsistent with the other two datasets.

In Experiment Pre-trained Classification, we observed the classification results from pre-trained models and collected latent representations using pre-trained models Wav2Vec 2.0 and ResNet50. We will discuss the results of the classification using pre-trained models in the next section.

4.2.2 Discussion For Experiment Pre-trained Classification

Our results showed that the Wav2Vec 2.0 model requires input specifications for window sizes 4, 40, and 100, also displayed in the line plots of G-Mean scores in Figure 4.3. Following the input specifications for the Wav2Vec 2.0 model, to generate accurate classifications, the input data must be fed into the model with window sizes larger than 400.

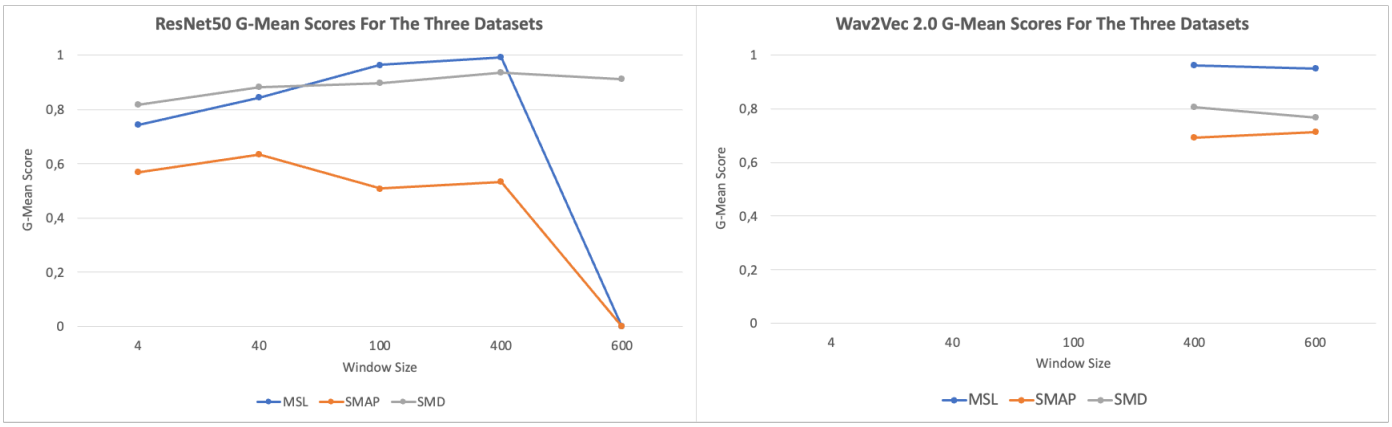


Figure 4.3: G-Mean Scores Derived Using Prototypical Network Approach For Pre-trained Models

On the other hand, the ResNet50 model generally outperformed the Wav2Vec 2.0 model across all window sizes, except window sizes 400 and 600 for the SMAP dataset. This suggests that the ResNet50 model may be better suited to fault detection classification due to its ability to capture more complex patterns in the data with larger window sizes considering the CNN architecture of the model.

Interestingly, we observed that the performance of the Wav2Vec 2.0 model increased when the window size increased from 400 to 600, except for the SMAP dataset. In situations where there is insufficient data, the Wav2Vec 2.0 model can be used instead of the ResNet50 model, as the G-Mean results of the ResNet50 model for window size 600 for the SMAP and MSL datasets are

0 as shown in Figure 4.3. The reason behind the classification performance of the Wav2Vec 2.0 model for window size 600 is that the model has been developed for speech recognition purposes, the model can capture more complex long-term patterns in data with larger window sizes, thus improving its fault classification performance.

The performance of the ResNet50 model varied depending on the dataset and window size. For the MSL and SMD datasets, the G-Mean Score generally increased as the window size increased to 400, then dropped or slightly decreased when the window size increased to 600. This indicates that for the ResNet50 model, there could be a window size that optimally balances the ability to capture intricate patterns within the data while maintaining high fault classification performance. However, with respect to the results for MSL and SMAP, due to the class distributions of representations, the evidence was insufficient to conclude a relationship between the larger window size and the classification performance (for class distributions, see Figure 4.4).

For the SMAP dataset, which has the least training data, the ResNet50 model's performance varied more noticeably with window size changes. The G-Mean Score increased with a window size of 40, lowered at 100, increased again to 400, and decreased dramatically to 0 at 600. This implies that when data is limited, ResNet50's performance may be more sensitive to window size changes. In contrast, with the SMD dataset, which contains the most training data, ResNet50 showed consistent performance across different window sizes, demonstrating its effectiveness with larger datasets.

We observed that pre-trained models can provide considerable classification performance comparable to state-of-the-art models. Therefore, pre-trained models can be a viable option for time series classification tasks, potentially offering a balance between performance and computational efficiency. Furthermore, our study revealed that when using the Prototypical Network approach, we need a sufficient amount of data to create clusters. In particular, we did not observe clusters when we used the Prototypical Network approach for the classification of ResNet50. However, based on curve trends and the classification results of the SMD dataset, we observed that the ResNet50 model offers comparable classification performance for larger window sizes when sufficient data are available.

In conclusion, our study highlights the essential role of window size in the classification of time series data. Our findings suggest that careful consideration should be given to the window size and model choice for classification tasks using pre-trained models. Therefore, we were able to find answers to the second research question.

4.3 Experiment Representation Quality Evaluation

In Experiment Representation Quality Evaluation, our primary focus is to track the impact of windowing on latent representation qualities by evaluating the results of quantified centroid-based evaluation metrics. In the following sections, we provide the centroid-based evaluation metrics results in a table. Here, metrics' results are shared for four classification models across five different window sizes along with their class distributions. As we described our metrics in Chapter 3, for the Minimum Inter-Centroid Distance metric, higher results indicate high separability, thus better representation qualities. On the other hand, for the Maximum Standard Deviation of Intra-Centroid Distances metric, lower values indicate isolated islands, thus better representation qualities. First, we share our observations from the results of Experiment Representation Quality Evaluation. Later in the discussion section, we provide four line plots of every model, where plots show their G-Mean scores. In these plots, we compared the metrics' scores to assess representation qualities for all classification models. In addition to this, our analysis is also visually supported by utilizing dimensionality reduction plots to visualize representations and validate the results and observations from centroid-based quantitative evaluation metrics. Finally, we discuss our findings

from the results of Experiment Representation Quality Evaluation.

In Experiment Representation Quality Evaluation, we defined our study to find an answer to the third research question:

RQ3) What is the impact of window size on latent representation qualities? Can we evaluate latent representation qualities using dimensionality reduction techniques and quantitative evaluation metrics?

4.3.1 Results For Experiment Representation Quality Evaluation

After finishing Experiment SOTA Classification and Experiment Pre-trained Classification, we tested centroid-based metrics to evaluate the representation qualities generated by SOTA and pre-trained models. According to the results shown in Figure 4.4, we share our observations below by first grouping metrics' results for all three datasets, later providing their corresponding results for SOTA and pre-trained models.

For the MSL dataset, we observed:

- MAD-GAN: the Inter-Centroid Distance increases significantly as the window size increases from 4 to 600. the Intra-Centroid Maximum Standard Deviation of Distances also increases as the window size increases from 4 to 600.
- OmniAnomaly: Both the Inter-Centroid Distance and the Intra-Centroid Maximum Standard Deviation of Distances decrease as the window size increases from 4 to 600.
- ResNet50: the Inter-Centroid Distance increases slightly as the window size increases from 4 to 100. Then, starts to slightly decrease from 100 to 400, and sharply drops from 400 to 600. the Intra-Centroid Maximum Standard Deviation of Distances increases from 4 to 100, decreases from 100 to 400, and drastically decreases from 400 to 600.

For the SMD dataset, we observed:

- MAD-GAN: the Inter-Centroid Distance increases as the window size increases from 4 to 600. The Intra-Centroid Maximum Standard Deviation of Distances increases slightly as the window size increases from 4 to 600.
- OmniAnomaly and ResNet50: Both models display similar trends for the Inter-Centroid Distance, and the metric value decreases as the window size increases from 4 to 600, with the exception that the metric value increases for the ResNet50 model when the window size increases from 100 to 400, then sharply decreases from 400 to 600.
- ResNet50: the Intra-Centroid Maximum Standard Deviation of Distances decreases as the window size increases from 4 to 600.
- OmniAnomaly: the Intra-Centroid Maximum Standard Deviation of Distances increases as the window size increases from 4 to 40, then slightly decreases from 40 to 600, and the metric value saturates after window size 400.

For the SMAP dataset, we observed:

- MAD-GAN: the Inter-Centroid Distance (Min) increases significantly as the window size increases from 4 to 40. Then, the metric value starts to drop sharply when the window size changes from 40 to 100; later, it gradually increases as the window size increases from 100 to 600. The Intra-Centroid Maximum Standard Deviation of Distances also increases as the window size increases from 4 to 100, then gradually decreases until 600.
- OmniAnomaly: Both the Minimum Inter-Centroid Distance and the Intra-Centroid Maximum Standard Deviation of Distances increase as the window size increases from 4 to 100. Then, the metric value gradually decreases as the window size increases from 100 to 600 for the Minimum Inter-Centroid Distance metric. For the Intra-Centroid Maximum Standard

		window_size = 4				window_size = 40				window_size = 100				window_size = 400				window_size = 600			
		number_of_test_wins(6697)			(607)	number_of_test_wins(6061)			(571)	number_of_test_wins(6001)			(511)	number_of_test_wins(5701)			(211)	number_of_test_wins(5501)			(11)
Dataset	Centroid-Based Metrics	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50
MSL Dataset	Inter Centroid - Distance	658,3522	319,7	-	8,2192	2711,0339	252,3679	-	10,0859	3701,7441	190,2468	-	20,4625	19310,6987	95,1685	4,5652	19,6924	34331,638	762,2429	4,618	0
	Intra-Centroid - Maximum Standard Deviation of Distances	1502,0161	1962,382(6)	-	3,3003(0)	2659,0692(0)	1770,6024(1)	-	4,5184(1)	3249,6892(0)	1631,9111(1)	-	4,8833(1)	15595,328(0)	1304,7810(1)	1,3361(0)	3,0961(1)	25674,0625(0)	1182,4(0)	1,3638(0)	0,1398(1)
	The Distribution of Representations {Anomaly(1)/ Non-Anomaly(0) Samples}	{0: 5964, 1: 133}	{0: 5964, 1: 133}	{0: 5964, 1: 133}	{0: 591, 1: 16}	{0: 5892, 1: 169}	{0: 5892, 1: 169}	{0: 5892, 1: 169}	{0: 519, 1: 52}	{0: 5772, 1: 229}	{0: 5772, 1: 229}	{0: 5772, 1: 229}	{0: 399, 1: 112}	{0: 5172, 1: 529}	{0: 5172, 1: 529}	{0: 5172, 1: 529}	{0: 77, 1: 134}	{0: 4772, 1: 729}	{0: 4772, 1: 729}	{0: 4772, 1: 729}	{1: 11}
		number_of_test_wins(8837)			(861)	number_of_test_wins(8601)			(825)	number_of_test_wins(8541)			(765)	number_of_test_wins(8241)			(465)	number_of_test_wins(8041)			(265)
	Model Names ->	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50
SMAP Dataset	Inter Centroid - Distance	3,4962	5,0709	-	2,6659	18,684	5,1595	-	2,3546	3,8142	5,3146	-	4,691	4,8183	3,4095	1,2593	9,73	7,959	0,4225	1,3148	0
	Intra-Centroid - Maximum Standard Deviation of Distances	188,3865(0)	267,6586(0)	-	3,1322(0)	252,8899(0)	269,7845(0)	-	2,8080(0)	223,3140(0)	273,7997(0)	-	6,0064(0)	178,5060(0)	282,9935(0)	1,6582(1)	5,8895(1)	159,4(1)	281,2926(1)	1,1567(1)	3,8080(1)
	The Distribution of Representations {Anomaly(1)/ Non-Anomaly(0) Samples}	{0: 8367, 1: 270}	{0: 8367, 1: 270}	{0: 8367, 1: 270}	{0: 826, 1: 35}	{0: 8223, 1: 378}	{0: 8223, 1: 378}	{0: 8223, 1: 378}	{0: 700, 1: 125}	{0: 7983, 1: 558}	{0: 7983, 1: 558}	{0: 7983, 1: 558}	{0: 532, 1: 233}	{0: 6992, 1: 1249}	{0: 6992, 1: 1249}	{0: 6992, 1: 1249}	{0: 136, 1: 329}	{0: 6392, 1: 1649}	{0: 6392, 1: 1649}	{0: 6392, 1: 1649}	{1: 265}
		number_of_test_wins(28476)			(1137)	number_of_test_wins(28440)			(1101)	number_of_test_wins(28380)			(1041)	number_of_test_wins(28080)			(741)	number_of_test_wins(27880)			(541)
SMD Dataset	Model Names ->	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50	MAD_GAN	OmniAnomaly	Wav2Vec	ResNet50
	Inter Centroid - Distance (Min)	45,5112	0,3975	-	5,7926	113,5303	0,3778	-	6,1063	297,9422	0,3347	-	6,0127	306,6567	0,2212	1,3976	7,6473	377,6911	0,1935	1,2793	4,1592
	Intra-Centroid - Maximum Standard Deviation of Distances	53,2649(1)	0,2061(1)	-	2,8119(1)	117,1096(1)	0,2556(1)	-	2,0071(1)	336,6113(1)	0,2841(1)	-	1,1962(1)	624,5444(1)	0,2732(1)	0,8883(1)	0,8040(1)	727,7203(1)	0,2706(1)	0,9368(1)	0,5740(1)
	The Distribution of Representations {Anomaly(1)/ Non-Anomaly(0) Samples}	{0: 25785, 1: 2691}	{0: 25785, 1: 2691}	{0: 25785, 1: 2691}	{0: 1015, 1: 122}	{0: 25461, 1: 2979}	{0: 25461, 1: 2979}	{0: 25461, 1: 2979}	{0: 847, 1: 254}	{0: 24921, 1: 3459}	{0: 24921, 1: 3459}	{0: 24921, 1: 3459}	{0: 727, 1: 314}	{0: 22221, 1: 5859}	{0: 22221, 1: 5859}	{0: 22221, 1: 5859}	{0: 234, 1: 507}	{0: 20470, 1: 7410}	{0: 20470, 1: 7410}	{0: 20470, 1: 7410}	{0: 34, 1: 507}

Figure 4.4: Comparison of Window-Based Representation Qualities for Wav2Vec 2.0, ResNet50, OmniAnomaly, and MAD-GAN Models ((0) Indicates Non-anomaly Cluster, (1) Indicates Anomaly Cluster)

Deviation of Distances metric, it first increases from 100 to 400, then slightly decreases from 400 to 600.

- ResNet50: the Inter-Centroid Distance (Min) increases slightly as the window size increases from 4 to 400, then drops sharply from 400 to 600. The Intra-Centroid Maximum Standard Deviation of Distances decreases from 4 to 40, increases from 40 to 100, decreases from 100 to 400, and decreases sharply from 400 to 600.
- Specifically for Wav2Vec 2.0 model, we observed an increase in all metrics in all three datasets except for the SMD dataset, Minimum Inter-Centroid Distance metric, when we increased the window size from 400 to 600.

Consequently, the Experiment Representation Quality Evaluation gave us crucial observations about representation qualities. Based on the results of our evaluations metrics, we extracted the relationship between the window size and representation qualities. We discuss these observations in detail in the discussions.

4.3.2 Discussion For Experiment Representation Quality Evaluation

Our study examined the effect of varying window sizes on the quality of latent representations derived from several multivariate time series models, such as OmniAnomaly, MAD-GAN, Wav2Vec 2.0, and ResNet50. Employing novel centroid-based evaluation metrics allowed us to quantitatively evaluate the quality of latent representations.

In Figure 4.5, results are shown for the first quantitative centroid-based evaluation metrics, Minimum of Inter-Centroid Distances in line plots. Note that the line plots in Figure 4.5 are provided only to make comparisons of the trend of changes in window size. The Inter-Centroid Distances are normalized between 0 and 1 to easily plot trend curves of our first metric, the Minimum of Inter-Centroid Distances.

Moreover, in Figure 4.6, results are shown for the second quantitative centroid-based metric, the Maximum Standard Deviation of Intra-Centroid Distances in line plots. Note that the line plots in Figure 4.6 are provided only to make comparisons of the trend of changes in window size. The Intra-Centroid Distances are normalized between 0 and 1 to easily plot the trend curves of our second metric, the Maximum Standard Deviation of Intra-Centroid Distances.

During the evaluation of quantitative representation metrics, we explored the impact of window size on the latent representation qualities of four models (MAD-GAN, OmniAnomaly, ResNet50, and Wav2Vec 2.0) in three datasets (MSL, SMD, and SMAP). Our findings revealed significant variations in the latent representation qualities of these models with changes in window size.

For the MSL dataset, the MAD-GAN model exhibited an increase in both the Inter-Centroid Distance and the Intra-Centroid Maximum Standard Deviation of Distances with larger window sizes. This indicates a divergence in representation quality, with improved separability but increased variability within clusters. The OmniAnomaly model, on the other hand, showed a decrease in these metrics, suggesting a struggle with larger window sizes. The ResNet50 model demonstrated nuanced performance, with an initial increase in Inter-Centroid Distance up to a window size of 100, followed by a decrease, suggesting an optimal window size for capturing complex patterns.

In the SMD dataset, the MAD-GAN model showed a consistent increase in the Inter-Centroid Distance metric with larger window sizes, indicating improved separability of representations. However, the Intra-Centroid Maximum Standard Deviation of Distances metric only increased slightly, suggesting a slight improvement in representation consistency. Both the OmniAnomaly and ResNet50 models showed a decrease in the Inter-Centroid Distance metric as the window size increased, suggesting a struggle to generate distinct representations for larger window sizes. The ResNet50 model, however, showed a consistent decrease in the Intra-Centroid Maximum Standard Deviation of Distances, indicating improved representation consistency.

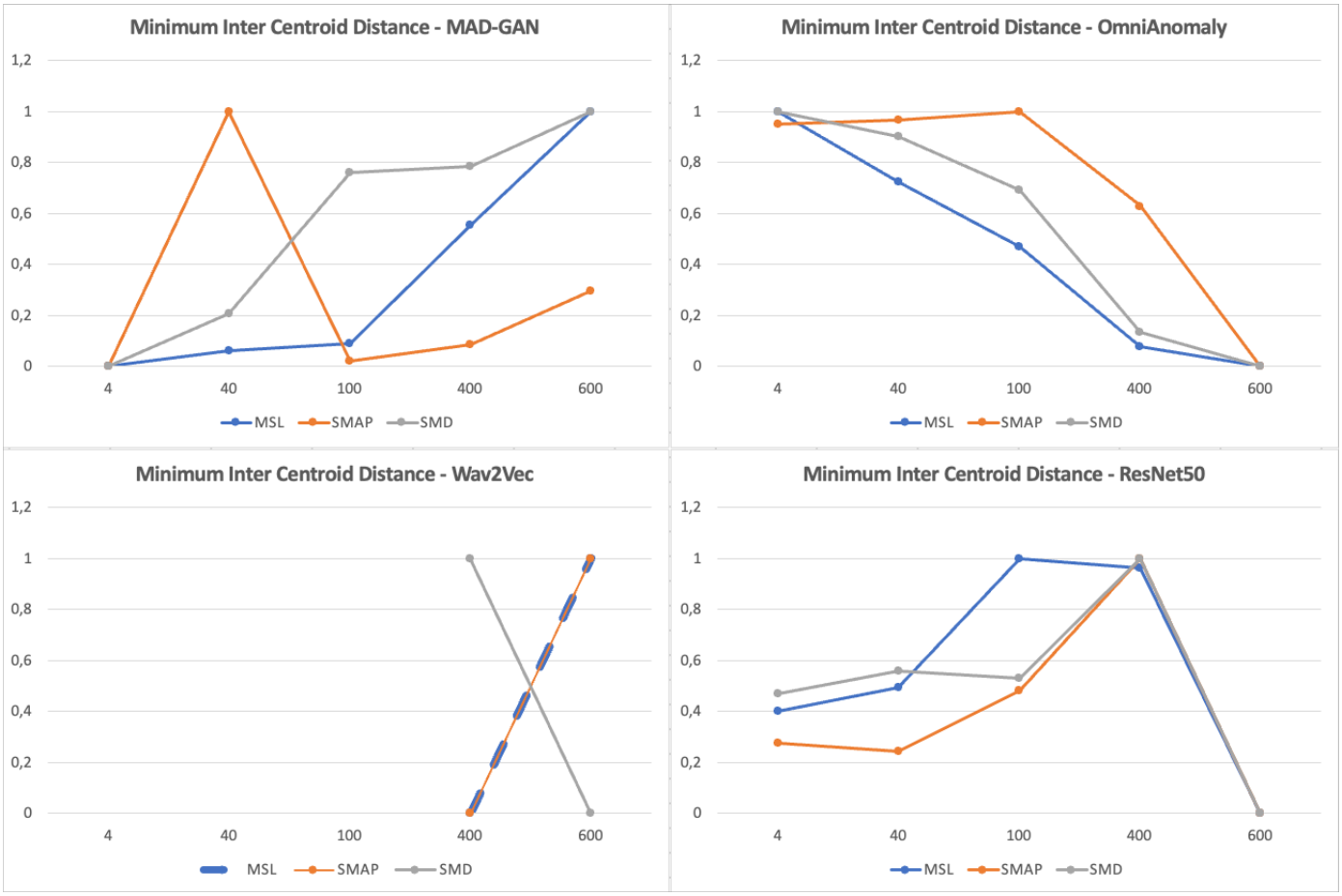


Figure 4.5: Minimum of Inter-Centroid Distances Metric Comparison for All Models

For the SMAP dataset, which has the smallest amount of training data, the MAD-GAN model's latent representation quality was more sensitive to changes in window size. The MAD-GAN model's Minimum Inter-Centroid Distance metric initially increased with window size, dropped sharply at a window size of 100, and gradually increased again. The suggested behavior indicates a complex relationship between window size and representation quality, in other words, when data is limited, the choice of window size should be carefully made. The OmniAnomaly model showed a similar trend, while the ResNet50 model required sufficient data for window sizes larger than 400.

Interestingly, the Wav2Vec 2.0 model showed an increase in all metrics in all three datasets, except the SMD data set, when the window size increased from 400 to 600. This indicates that the Wav2Vec 2.0 model may perform better with larger window sizes, particularly when there is a high degree of overlap between consecutive windows.

Our findings suggest that one should carefully choose a window size and model when dealing with overlapping values in time series data.

As supporting evidence for the quantitative evaluations, we utilized dimensionality reduction techniques to visualize the representation behavior generated by these models under different window sizes in 2D. Later, assess their performance by visually validating our findings from quantitative representation evaluation metrics.

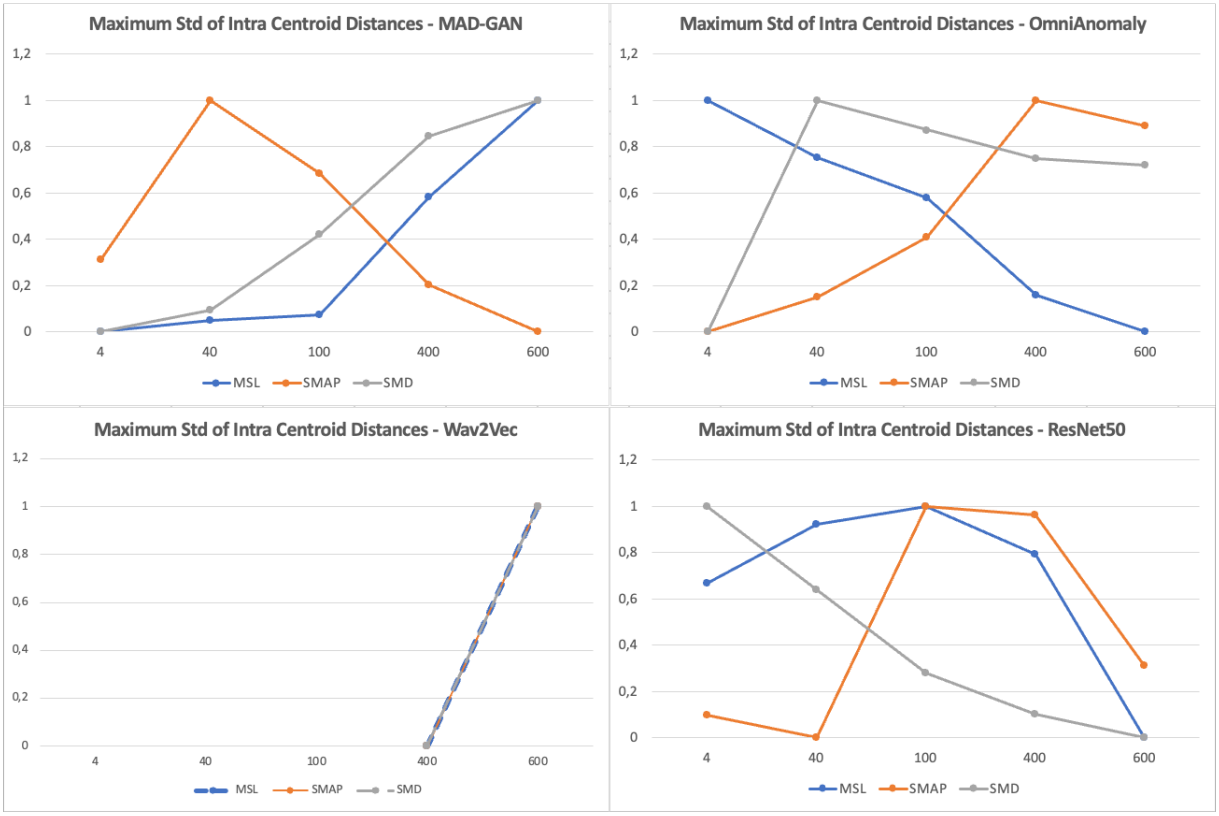


Figure 4.6: Maximum Standard Deviation of Intra-Centroid Distances Metric Comparison for All Models

4.3.3 Visually Evaluating Representation Qualities

In this section, we will share the results of the dimensionality reduction plots generated using t-SNE to visualize the latent representations generated from multivariate time series classification models. We intended to use t-SNE plots to visually validate the results we derived from our novel quantitative metrics to evaluate representations.

In the following, we share representation plots that were generated using t-SNE for all three datasets, ranging from window sizes 4 to 600. As shown in every plot, when the window size increased, we observed fewer isolated "islands" (clustered points) and blended representation visualizations. Considering the higher rate of overlapping values for larger windows, as the number of similar data points within each window increases, we expect to see similar patterns between windows. Therefore, for larger windows, we expect to see blended dimensionality reduction plots, as shown in Figure 4.12. In this visualization, we can visually validate how we can use the Minimum Inter-Centroid Distance metric to evaluate the quality of representations.

On the other hand, we observed that for smaller window sizes, the rate of overlapping values within smaller windows is lower, and windows are less similar. Therefore, the less similar windows can be easily separated. In a plot where the representations are less similar, we expect to see isolated islands, indicating a distinct separation of the window-based representations, as shown in Figure 4.8. Those isolated islands are what we intended to see in quality representations. Therefore, in this visualization, we can visually validate the rationale behind the Maximum of Intra-Centroid Distances metric.

Moreover, in the t-SNE plots, we observed fewer data points in smaller window sizes because

similar values mapped to the same dimensions when using the t-SNE method.

In conclusion, using the t-SNE method, we also visually analyzed the quality of the representations and validated the results from quantitative metrics. We found an answer to the third research question. Additionally, considering the various results we derived and discussed earlier, we presented the challenge of finding an optimal window size that both provides quality representations and better classification results. Based on our results and the discussion for Experiment Representation Quality Evaluation, we concluded that larger windows do not always guarantee quality representations. Therefore, considering the classification results of both SOTA and pre-trained models, discussions from Experiment SOTA Classification and Experiment Pre-trained Classification, and evaluations of representation qualities in Experiment Representation Quality Evaluation, we found a trade-off relationship between the representation quality and the classification results in terms of window sizes.

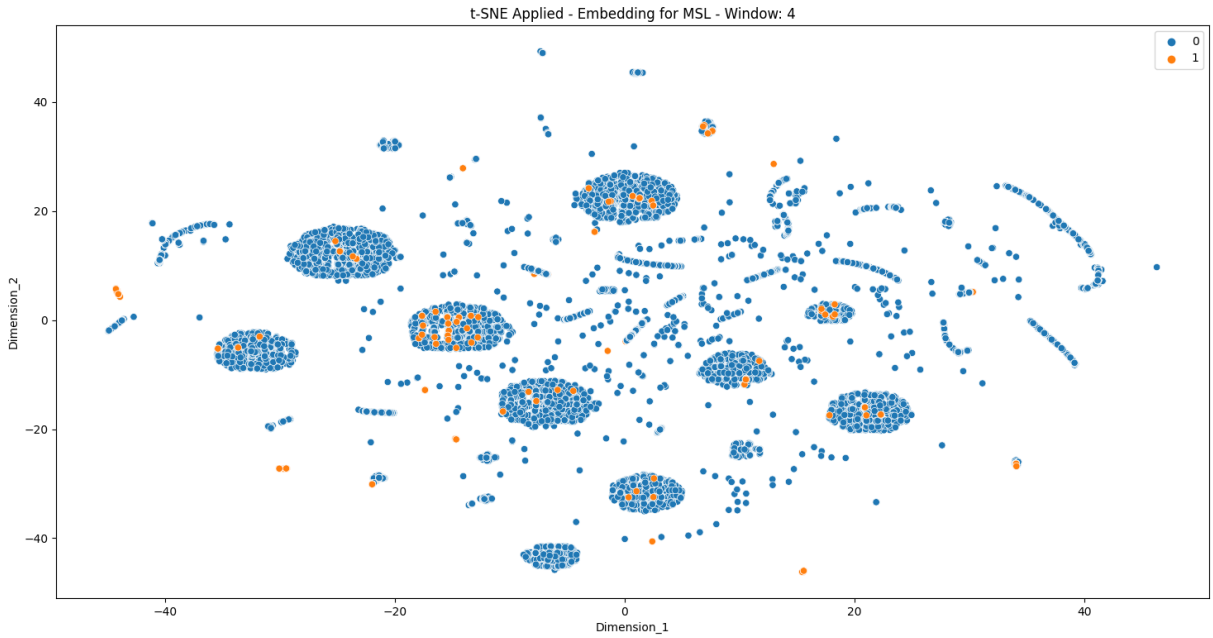


Figure 4.7: t-SNE Plot for Latent Representations of OmniAnomaly Model on MSL Dataset with Window Size 4

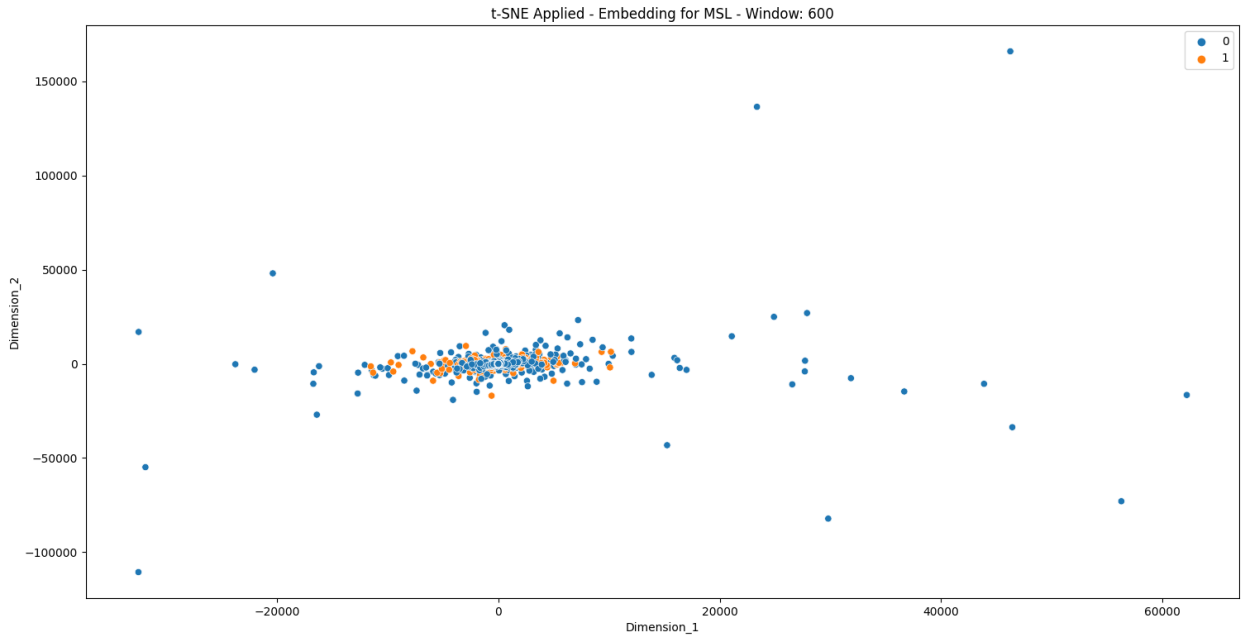


Figure 4.8: t-SNE Plot for Latent Representations of OmniAnomaly Model on MSL Dataset with Window Size 600

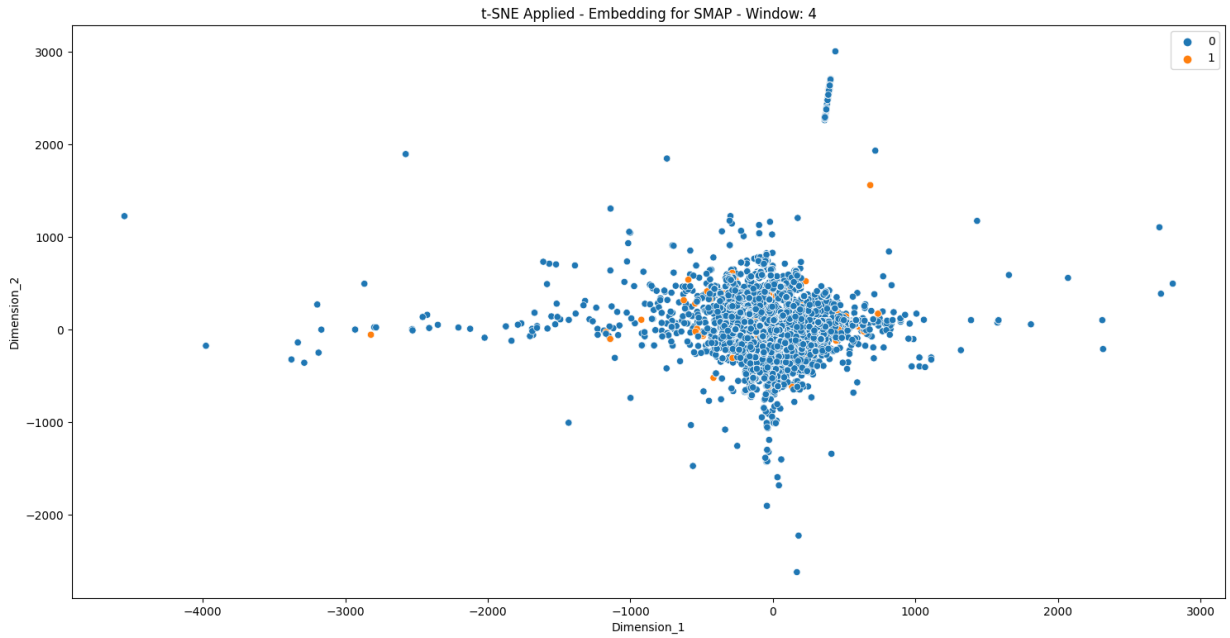


Figure 4.9: t-SNE Plot for Latent Representations of OmniAnomaly Model on SMAP Dataset with Window Size 4

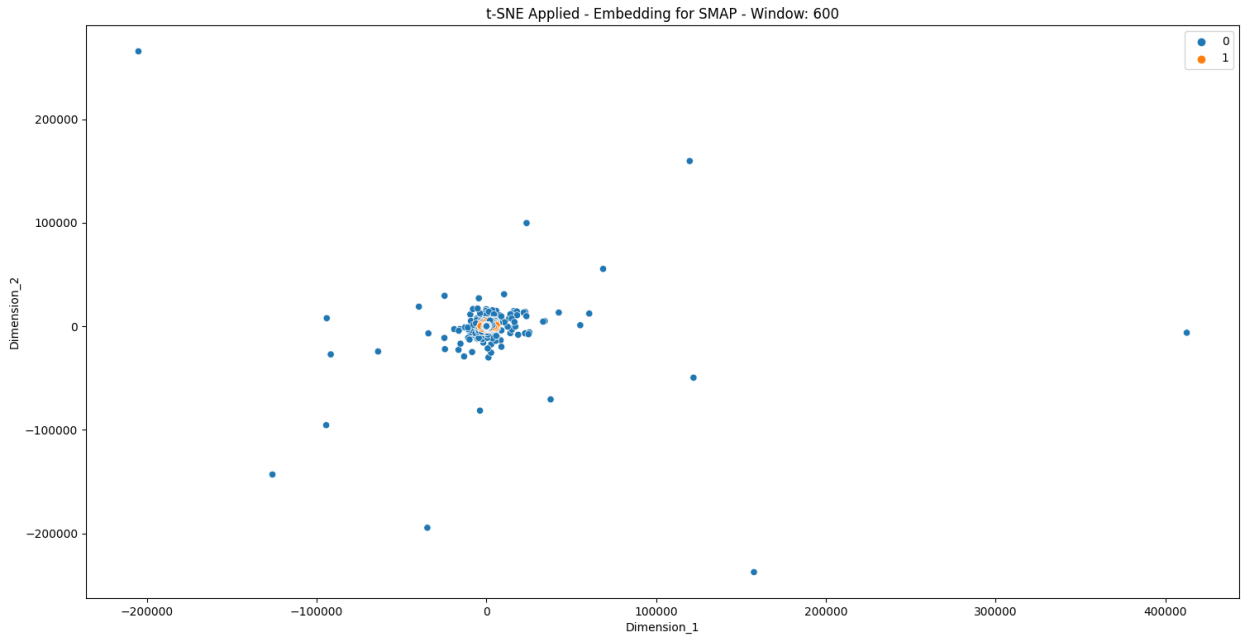


Figure 4.10: t-SNE Plot for Latent Representations of OmniAnomaly Model on SMAP Dataset with Window Size 600

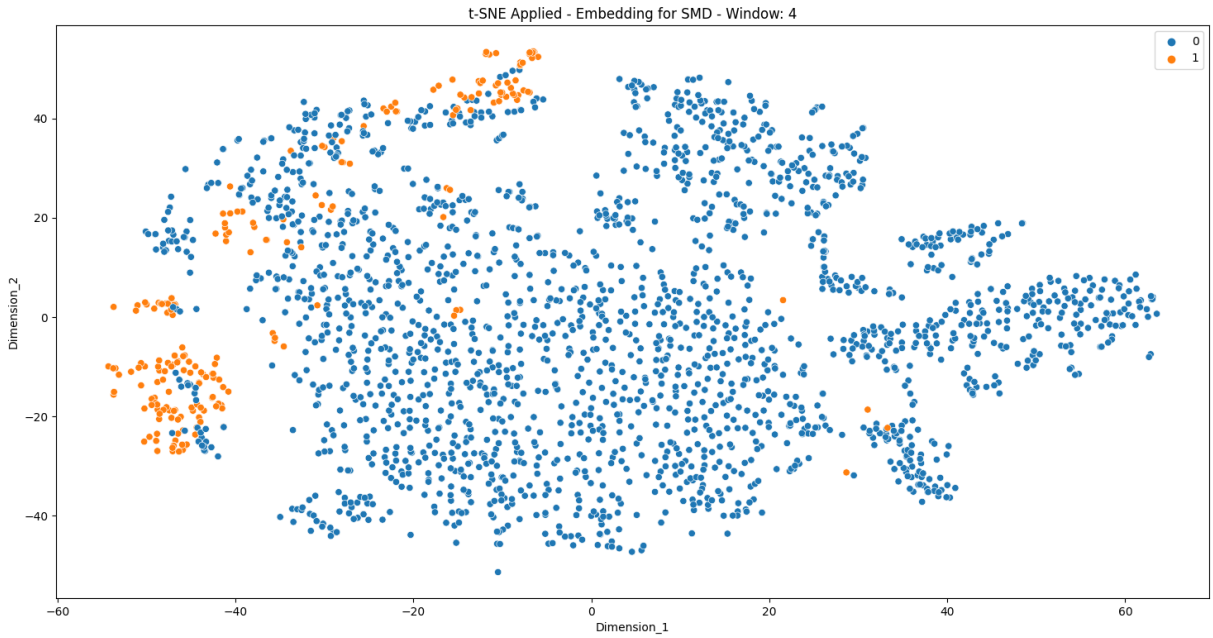


Figure 4.11: t-SNE Plot for Latent Representations of OmniAnomaly Model on SMD Dataset with Window Size 4

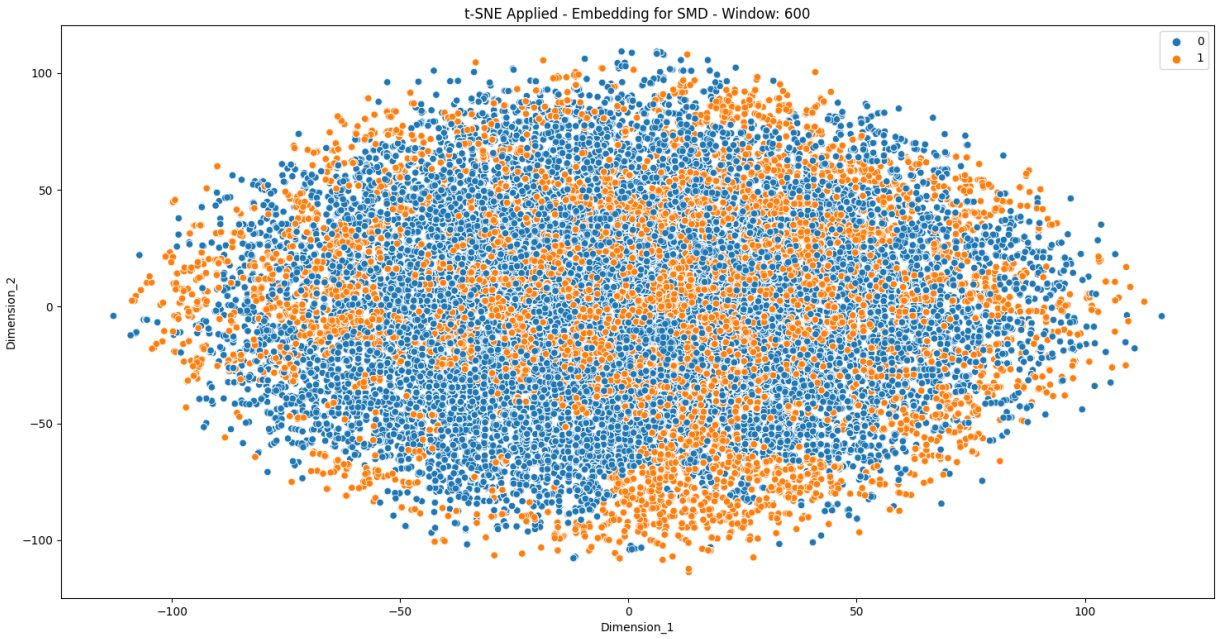


Figure 4.12: t-SNE Plot for Latent Representations of OmniAnomaly Model on SMD Dataset with Window Size 600

Chapter 5

Conclusions

Time series representation is a complex task yet brings immense advantages with an effective representation model. The use of an effective time series representation model can be for fault detection, anomaly detection, predictive maintenance, and so on. Our focus is on analyzing the impact of window size on classification and latent representation quality of time series deep learning models such as Wav2Vec (Pre-trained), ResNet50 (Pre-trained), OmniAnomaly (Flow-Based Variational Autoencoder Representation Framework), and MAD-GAN (GAN model). Our study has shown that it is important to pick an adequate window size to capture faults in time series data.

During our experiments with pre-trained models, we saw the potential to use pre-trained models for multivariate time series fault classification. For faster prototyping and lower training costs, pre-trained models, such as Wav2Vec 2.0 and ResNet50, are suggested to be considered. We can use these pre-trained models as baselines to evaluate fault classification performance and representation qualities. For existing SOTA models, we observed OmniAnomaly as a robust classification model that outputs consistent results.

Regarding the impact of window size, we expected to see the highest classification result for an optimal window size. However, we found that the classification results indicated that the larger the window size, the better the classification results for the distinct window sizes we used in our experiment. However, larger window sizes do not always guarantee quality representations in terms of our centroid-based metrics.

Unlike the OmniAnomaly model, the MAD-GAN model requires in-depth tuning of hyper-parameters. Moreover, we observed mode collapse during training the MAD-GAN model with distinct window sizes. Therefore, we were unable to formulate the relationship between classification performance and the size of the window used in the experiment for the MAD-GAN model. To use the MAD-GAN model in practice, we require substantial experiments to evaluate its performance.

Finally, as our contribution to academia, our experiments, specifically the classification results and the representation quality evaluation results, have shown that there is a trade-off relationship between the classification performances and representation qualities, similar to the bias-variance trade-off when we change the size of windows. For larger window sizes, while achieving higher classification results, one can give up latent representation qualities. Therefore, the trade-off indicates the relationship between the size of the window and the classification performance (and representation quality). Our extensive experiments using three public datasets have laid the groundwork for a more comprehensive understanding of the role of window size in time series representation. Our findings can be used to better understand the representations generated from the classification models and their classification performances. These findings can help us assess

the importance of choosing the proper window size for tasks such as fault detection, anomaly detection, and predictive maintenance.

5.1 Limitations and Future Work

5.1.1 Limitations

Throughout our research, we encountered several limitations that we acknowledge. Firstly, the MAD-GAN model we utilized exhibited sensitivity to hyper-parameter tuning. This sensitivity not only posed challenges in obtaining consistent results but also led to the mode collapse issue, complicating the training process. Additionally, the performance of the ResNet50 model was notably dependent on the volume of training data, particularly for datasets with limited data, such as the SMAP dataset. Our study also sheds light on the influence of window size on classification and representation quality. However, the ideal window size might differ across datasets and application contexts. We introduced novel centroid-based evaluation metrics, which, while promising, are still in their early stages. The metrics we introduced can be advanced to offer a more holistic evaluation of the representation qualities. The computational demands associated with training models such as MAD-GAN, given the extensive hyper-parameter tuning required, could make them less suitable for real-time applications.

5.1.2 Future Work

Our research paved the way for several possibilities for future work. One potential direction is the development of an optimization algorithm to automatically determine the optimal window size for various datasets and models based on the quality measurements we define. This would optimize the process of selecting the most effective window size, ensuring a balance between classification performance and representation quality. Addressing the mode collapse issue in GANs is another area open for exploration, possibly by investigating alternative training strategies or model architectures. To enhance the generalizability of our findings, future research could cover a broader range of datasets, especially those with varying characteristics and complexities. Our centroid-based evaluation metrics also present opportunities for further refinement and validation, aiming to establish a more robust framework for evaluating time series representation. Given the promising results observed with pre-trained models such as Wav2Vec 2.0 and ResNet50, it would be beneficial to explore other pre-trained models for time series classification tasks. Finally, applying our findings to real-world contexts, such as predictive maintenance and anomaly detection, provides insights into the applicability and efficacy of these models in industrial applications.

Bibliography

[1] Eit digital. https://masterschool.eitdigital.eu/data-science. 2

[2] Huggingface. https://huggingface.co/docs/transformers/model_doc/wav2vec2. 30

[3] Introducing multivariate anomaly detection. https://techcommunity.microsoft.com/t5/ai-cognitive-services-blog/introducing-multivariate-anomaly-detection/ba-p/2260679. 2

[4] Resnet50. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50. 15, 22, 30

[5] Stratifiedkfold sklearn. https://scikit-learn/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. 20, 31

[6] Wav2vec2. https://huggingface.co/docs/transformers/model_doc/wav2vec2. 15, 22, 30

[7] Over-fitting suppression training strategies for deep learning-based atrial fibrillation detection. https://pubmed.ncbi.nlm.nih.gov/33387183/, DOI=10.1007/s11517-020-02292-9, Jan 2021. 20

[8] Few-shot learning framework for fault classification and concept drift detection-2022, Feb 2022. 16

[9] Asml. https://www.asml.com/en, Jun 2023. 2

[10] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. 9, 15

[11] Taha Belkhouja, Yan Yan, and J. Doppa. Dynamic time warping based adversarial framework for time-series domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 3

[12] Gaudenz Boesch. Deep residual networks (resnet, resnet50) - 2023 guide - viso.ai, Jan 2023. 16

[13] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2015. 12

[14] P. Brockwell and R. Davis. Time series: Theory and methods. *Springer*, 2013. 5

[15] Jason Brownlee. Feature selection for machine learning in python, 2020. 19

[16] Guilherme A Campos, Wagner Meira Jr, and Robson L Carneiro. Time series forecasting with the long short-term memory network in python. *Neural Networks*, 98:296–312, 2018. 6

[17] Z. Chen, D. Chen, Z. Yuan, X. Cheng, and X. Zhang. Learning graph structures with transformer for multivariate time-series anomaly detection in iot. *IEEE Internet of Things Journal*, 2021. 6, 7

[18] Michael Christ, Nicolas Braun, Joris Neuffer, Andreas W Kempa-Liehr, and Claus Weihs. Tsfresh: A python package for automatic feature extraction from time series. *arXiv preprint arXiv:1610.07717*, 2018. 7

[19] A. Deng and B. Hooi. Graph neural network-based anomaly detection in multivariate time series. *AAAI*, 2021. 7

[20] Kaize Ding, Qinghai Zhou, Hanghang Tong, and Huan Liu. Few-shot network anomaly detection via cross-network meta-learning. 2021. 22

[21] Johann Faouzi and Hicham Janati. pyts: A python package for time series classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020. 7

[22] Md Fazle Karim, Somshubra Shukla, Sanjay Raman, and Apalak Khatua. Detsec: A deep learning-based framework for time series classification. In *International Conference on Database and Expert Systems Applications*, pages 287–302. Springer, 2020. 8

[23] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing, 2019. 12

[24] Ankush Ganguly and Samuel W. F. Earp. An introduction to variational inference, 2021. 11, 12

[25] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 13

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 9, 15

[27] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: Yet another way to carve up the variational evidence lower bound. 12

[28] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '18, page 387–395, New York, NY, USA, 2018. Association for Computing Machinery. 17, 18

[29] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. 19

[30] Vijaya Chandra Jadala, Sai Kiran Pasupuletti, S. Hrushikesava Raju, S. Kavitha, Ch.M H Sai Bhaba, and B. Sreedhar. Need of intenet of things, industrial iot, industry 4.0 and integration of cloud for industrial revolution. In *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–5, 2021. 1

[31] M. Khorasani, J. Loy, A.H. Ghasemi, E. Sharabian, M. Leary, H. Mirafzal, P. Cochrane, B. Rolfe, and I. Gibson. A review of industry 4.0 and additive manufacturing synergy. *Rapid Prototyping Journal*, 28(8):1462–1475, 2022. 1

[32] Jihoon Kim et al. Fault detection for photovoltaic systems using multivariate analysis with electrical and environmental variables. *IEEE Journal of Photovoltaics*, 2020. 8

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 11

[34] Lattawit Kulanuwat, Chantana Chantrapornchai, Montri Maleewong, Papis Wongchaisuwat, Supaluk Wimala, Kanoksri Sarinnapakorn, and Surajate Boonya-aroonnet. Anomaly detection using a sliding window technique and data imputation with machine learning for hydrological time series. *Water*, 13(13), 2021. 1

[35] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, 2019. 9, 14

[36] Hailin Li, Chunpei Lin, Xiaoji Wan, and Zhengxin Li. Feature representation and similarity measure based on covariance sequence for multivariate time series. *IEEE Access*, 7:67018–67026, 2019. 1

[37] Hailin Li, Chunpei Lin, Xiaoji Wan, and Zhengxin Li. Feature representation and similarity measure based on covariance sequence for multivariate time series. *IEEE Access*, 2019. 6

[38] Raymond Liu and Duncan F. Gillies. Overfitting in linear feature extraction for classification of high-dimensional image data. *Pattern Recognition*, 53:73–86, 2016. 8

[39] Krisha Mehta. Measuring mode collapse in gans using weights biases, Jul 2020. 30

[40] Andreas C Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016. 19

[41] Minh Nguyen. m-tsne: A framework for visualizing high-dimensional multivariate time series. *IEEE*, 2017. 3

[42] Rajesh Pandit, Kyungmoo Kim, and Ying Tan. Operational variables for improving industrial wind turbine yaw misalignment early fault detection capabilities using data-driven techniques. *IEEE Transactions on Industrial Informatics*, 2021. 8

[43] Author1 Reunanen, Author2, Author3, Author4, and Author5. Unsupervised online detection and prediction of outliers in streams of sensor data. *International Journal of Data Science and Analytics*, 2019. 23

[44] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. 12

[45] Luca Rosafalco, Andrea Manzoni, Alberto Corigliano, and Stefano Mariani. A time series autoencoder for load identification via dimensionality reduction of sensor recordings. *Engineering Proceedings*, 2(1), 2020. 1, 3

[46] S. M. S. Salahin, M. D. S. Ullaa, Saif Ahmed, Nabeel Mohammed, Taseef Hasan Farook, and J. Dudley. One-stage methods of computer vision object detection to classify carious lesions from smartphone imaging. *Oral*, 3(2):16, 2023. 10

[47] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouët. Anomaly detection in streams with extreme value theory. pages 1067–1075, 08 2017. 13

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 16

[49] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017. 22, 25

[50] Harald Steck. Autoencoders that don't overfit towards the identity. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 8

[51] Stephen M. Stigler. Thomas bayes's bayesian inference. *Journal of the Royal Statistical Society. Series A (General)*, 145(2):250–258, 1982. 11

[52] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery. 9, 12, 17, 18

[53] Nuo Tan et al. Optimal robust fault detection of discrete-time lpv systems with measurement error-affected scheduling variables combining zkf and pqp. *International Journal of Robust and Nonlinear Control*, 2020. 8

[54] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data, 2022. 26

[55] Ilias P. Vlachos, Rodrigo Martinez Pascazzi, George Zobolas, Panagiotis Repoussis, and Mihalis Giannakis. Lean manufacturing systems in the area of industry 4.0: a lean automation plan of agvs/iot integration. *Production Planning & Control*, 34(4):345–358, 2023. 1

[56] Xiang Wang, Xiaoyong Li, Junxing Zhu, Zichen Xu, Kaijun Ren, Weiming Zhang, Xinwang Liu, and Kui Yu. A local similarity-preserving framework for nonlinear dimensionality reduction with neural networks, 2021. 22

[57] Parul Yadav, Manish Gaur, Nishat Fatima, and Saqib Sarwar. Qualitative and quantitative evaluation of multivariate time-series synthetic data generated using mts-tgan: A novel approach. *Applied Sciences*, 2023. 3

[58] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Multivariate time-series anomaly detection via graph attention network. In *ICDM*, 2020. 7

[59] P. Zhao, X. Chang, and M. Wang. A novel multivariate time-series anomaly detection approach using an unsupervised deep neural network. *IEEE Access*, 2021. 6, 7

[60] Ming Zheng, Fei Wang, Xiaowen Hu, Yuhao Miao, Huo Cao, and Mingjing Tang. A method for analyzing the performance impact of imbalanced binary data on machine learning models, Nov 2022. 10

[61] Lei Zhou, Bolun Chen, Hu Liu, and Liuyang Wang. Personalized sliding window recommendation algorithm based on sequence alignment. 2022. 15, 16

Appendix A

Appendix

Models/ Window_Size	4_1	4_2	4_3	40_1	40_2	40_3	100_1	100_2	100_3	400_1	400_2	400_3	600_1	600_2	600_3
OmniAnomaly	wins: number of windows used														
MSL (Trained)	train_wins(4305) / test_wins(6097)			train_wins(4269) / test_wins(6061)			train_wins(4209) / test_wins(6001)			train_wins(3909) / test_wins(5701)			train_wins(3709) / test_wins(5501)		
Tested on MSL	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,8737	0,8712	0,8726	0,8728	0,8728	0,8724	0,8724	0,8721	0,8724	0,8651	0,865	0,865	0,8595	0,8597	0,8596
Recall (~)	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999
Specificity (~)	0,7633	0,759	0,7615	0,7619	0,7618	0,7611	0,7611	0,761	0,7613	0,7485	0,7483	0,7483	0,7389	0,7396	0,739
SMAP (Trained)	train_wins(2877) / test_wins(8637)			train_wins(2841) / test_wins(8601)			train_wins(2781) / test_wins(8541)			train_wins(2481) / test_wins(8241)			train_wins(2281) / test_wins(8041)		
Tested on SMAP	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,916	0,8966	0,9166	0,9165	0,9155	0,9173	0,9167	0,9161	0,9153	0,91637	0,9171	0,9176	0,91633	0,9166	0,9162
Recall (~)	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467	0,8467
Specificity (~)	0,9909	0,9496	0,9923	0,9922	0,9899	0,9937	0,9925	0,9913	0,9896	0,9917	0,9934	0,9944	0,9916	0,9922	0,9913
SMD (Trained)	train_wins(28476) / test_wins(28476)			train_wins(28440) / test_wins(28440)			train_wins(28380) / test_wins(28380)			train_wins(28080) / test_wins(28080)			train_wins(27880) / test_wins(27880)		
Tested on SMD	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,9799	0,98	0,9858	0,9903	0,9851	0,9882	0,9913	0,9926	0,9906	0,9946	0,9909	0,9916	0,9922	0,9921	0,992
Recall (~)	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999
Specificity (~)	0,9602	0,9612	0,9719	0,9807	0,9705	0,9766	0,9828	0,9853	0,9812	0,9894	0,9818	0,9834	0,9844	0,9844	0,9842

Figure A.1: OmniAnomaly Model's Approximated Experiment Results Ran Three Times With Different Seeds

Models/ Window_Size	4_1	4_2	4_3	40_1	40_2	40_3	100_1	100_2	100_3	400_1	400_2	400_3	600_1	600_2	600_3
MAD_GAN	wins: number of windows used														
MSL (Trained)	train_wins(4305) / test_wins(6097)			train_wins(4269) / test_wins(6061)			train_wins(4209) / test_wins(6001)			train_wins(3909) / test_wins(5701)			train_wins(3709) / test_wins(5501)		
Tested on MSL	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,8659	0,8661	0,866	0,8517	0,853	0,8524	0,8432	0,8445	0,8572	0,8068	0,8146	0,8457	0,7748	0,7642	0,7716
Recall (~)	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,999	0,999	0,9999	0,9999	0,9999	0,9999
Specificity (~)	0,7497	0,7503	0,7501	0,7255	0,7277	0,7266	0,7111	0,7133	0,7347	0,651	0,6636	0,7153	0,6004	0,584	0,5954
SMAP (Trained)	train_wins(2877) / test_wins(8637)			train_wins(2841) / test_wins(8601)			train_wins(2781) / test_wins(8541)			train_wins(2481) / test_wins(8241)			train_wins(2281) / test_wins(8041)		
Tested on SMAP	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,9126	0,9921	0,9123	0,9831	0,9864	0,9863	0,9828	0,9818	0,9807	0,8682	0,8692	0,8774	0,9524	0,9387	0,9333
Recall (~)	0,8467	0,9999	0,8467	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,8467	0,8467	0,8467	0,9999	0,9999	0,9999
Specificity (~)	0,9837	0,9843	0,983	0,9666	0,9731	0,9729	0,966	0,964	0,9618	0,8903	0,8923	0,9092	0,9071	0,8812	0,8701
SMD (Trained)	train_wins(28476) / test_wins(28476)			train_wins(28440) / test_wins(28440)			train_wins(28380) / test_wins(28380)			train_wins(28080) / test_wins(28080)			train_wins(27880) / test_wins(27880)		
Tested on SMD	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)	seed(1)	seed(1000)	seed(100000)
G - Mean (~)	0,4557	0,9949	0,9896	0,9983	0,9979	0,4557	0,9186	0,9901	0,9984	0,9185	0,9185	0,805	0,9168	0,8067	0,8174
Recall (~)	0,2077	0,9973	0,9973	0,9973	0,9973	0,2077	0,844	0,9973	0,9973	0,844	0,844	0,6494	0,844	0,6726	0,6726
Specificity (~)	1	0,9924	0,9819	0,9993	0,9986	1	0,9998	0,98228	0,9995	0,9997	0,9997	0,9978	0,996	0,9675	0,9933
	collapsed	used	used	used	used	collapsed									

Figure A.2: MAD-GAN Model's Approximated Experiment Results Ran Three Times With Different Seeds