ANOMALY DETECTION IN INDUSTRIAL PROCESSES USING CURRICULUM-BASED DEEP LEARNING

by

Devilliers Caleb Dube

B.S., Electrical and Electronics Engineering, Eastern Mediterranean University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2024

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Mehmet Akar for his invaluable guidance and insightful feedback throughout my research. I am thankful for his expertise and demonstration of patience, even during my most impatient moments.

I would like to thank Engineers Sencer Sultanoğlu and Mustafa Çom for their guidance in part of my thesis and Eliar Elektronik San. A.Ş., for sharing the textile data used in this thesis.

I also want to acknowledge my thesis defense committee, Prof. Murat Saraçlar and Prof. Fikret Çalışkan, for their valuable discussions and constructive feedback during the defense.

I would like to thank Türkiye Scholarships program for providing me with a fully-funded scholarship for my Master's degree. I also like to thank Prof. Çiğdem Eroğlu Erdem and Assoc. Prof. Ömer Korçak for giving me the opportunity to work on a research project funded by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). This project notably enhanced my scientific and research skills which were highly beneficial for my thesis.

A very special thank you to my mother, Gconiwe Ndhlovu and my sister, Faith Dube for their prayers, reassurance, and emotional support during my research. I am greatly indebted to my uncle, Assoc. Prof. Finex Ndhlovu and his family for their support throughout my academic pursuits. I also like to thank the rest of my family members and friends for their encouragement that gave me strength and determination to overcome the challenges I faced.

ABSTRACT

ANOMALY DETECTION IN INDUSTRIAL PROCESSES USING CURRICULUM-BASED DEEP LEARNING

Anomaly detection is increasingly drawing attention in research as anomalies encountered in the industry become commonplace and more complex. In industrial processes, high dimensional multivariate time series (MVTS) data is generated from sensors and actuators. To guarantee the safety of the equipment and high-quality products, anomalies recorded by these components have to be detected. In this thesis, the use of deep learning for industrial anomaly detection is studied. Particularly, curriculum learning (CL), a training paradigm for machine learning models is comprehensively analyzed. CL follows the learning progression of humans and animals, in which the model is first trained with "easy" data, and more "difficult" data is gradually introduced as training advances. This thesis proposes data-based CL for MVTS anomaly detection and introduces the CL concept to the learner (model). These frameworks are applied to two different case studies, first using a benchmark secure water treatment dataset and then a real-world textile dataset. In the data-based method, extensive experiments are conducted to determine the most important MVTS features for improved model performance. In the model-based framework, a simple Long Short-Term Memory Encoder-Decoder model is trained first, and then a complex Multi-Scale Convolutional Recurrent Encoder-Decoder model is utilized in the final training round. These techniques are evaluated based on the F1 score, with a data-based approach producing the best results among the proposed designs.

ÖZET

ENDÜSTRİYEL SÜREÇLERDE MÜFREDAT TABANLI DERİN ÖĞRENME KULLANARAK ANOMALİ TESPİTİ

Anomali tespiti, endüstride karşılaşılan anormalliklerin yaygınlaşması ve karmaşıklığının artmasıyla araştırmalarda giderek daha fazla dikkat çekmektedir. Endüstriyel süreçlerde, sensörler ve aktüatörlerden yüksek boyutlu çok değişkenli zaman serileri (ÇDZS) verileri üretilmektedir. Ekipmanın güvenliğini ve yüksek kaliteli ürünleri garantilemek için, bu bileşenler tarafından kaydedilen anormalliklerin tespit edilmesi gerekmektedir. Bu tezde, endüstriyel anormallik tespiti için derin öğrenme kullanımı çalışılmıştır. Özellikle, makine öğrenme modelleri için bir eğitim paradigması olan müfredat öğrenimi (MÖ) kapsamlı bir şekilde incelenmektedir. MÖ, insanların ve hayvanların öğrenme ilerlemesini izler, bu süreçte model önce "kolay" verilerle eğitilir ve eğitim ilerledikçe daha "zor" veriler aşamalı olarak tanıtılır. Bu tez, ÇDZS anormallik tespiti için veri tabanlı MÖ'yi önermekte ve MÖ kavramını model üzerinde incelemektedir. Bu yöntem, öncelikle literatürde mevcut bir güvenli su arıtma veri kümesi ve ardından gerçek dünya tekstil veri kümesi kullanılarak iki farklı vaka çalışmasına uygulanmıştır. Veri tabanlı yöntemde, model performansını artırmak için en önemli ÇDZS özelliklerini belirlemek için kapsamlı deneyler yapılmıştır. Model tabanlı çerçevede, önce basit bir Uzun Kısa Süreli Bellek Kodlayıcı-Çözücü modeli eğitilmiş ve ardından son eğitim turunda karmaşık bir Çok Ölçekli Evrişimsel Tekrarlayan Kodlayıcı-Çözücü modeli kullanılmıştır. Bu teknikler, önerilen tasarımlar arasında en iyi sonuçları üreten veri tabanlı bir yaklaşıma dayalı olarak F1 skoru ile değerlendirilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xiv
1. INTRODUCTION 1
1.1. Multivariate Time Series Anomaly Detection 2
1.2. Motivation of the Thesis 5
1.3. Contributions of the Thesis 8
1.4. Organization of the Thesis 10
2. OVERVIEW OF ANOMALY DETECTION TECHNIQUES 11
2.1. Probabilistic and Statistical Models 11
2.2. Finite State Machine Models 12
2.3. Machine Learning Models 15
2.3.1. Linear Models 15
2.3.2. Proximity-based Models 16
2.3.3. Automated Machine Learning Models 18
2.4. Deep Learning Models 19
2.4.1. Reconstruction-based Models 19
2.4.2. Forecasting-based Models 20
2.4.3. Generative Models 21
2.5. Summary of the Chapter 22
3. DEEP LEARNING METHODOLOGY AND EVALUATION METRICS . . 23
3.1. Machine Learning Overview 23
3.2. Multilayer Perceptrons 24
3.2.1. Activation Functions 25

3.3. Convolutional Neural Networks . . . 26
3.4. Recurrent Neural Networks . . . 28
3.4.1. Long Short-Term Memory . . . 29
3.5. Autoencoders . . . 30
3.6. Multi-Scale Convolutional Recurrent Encoder-Decoder Framework . . . 31
3.7. Curriculum Learning . . . 34
3.7.1. Curriculum Design . . . 35
3.7.2. Predefined Curriculum Learning . . . 36
3.7.2.1. Predefined Difficulty Measurers . . . 36
3.7.2.2. Predefined Training Schedulers . . . 37
3.7.3. Automatic Curriculum Learning . . . 37
3.8. Evaluation Metrics . . . 39
3.9. Summary of the Chapter . . . 40
4. ANOMALY DETECTION IN WATER TREATMENT PROCESSES . . . 41
4.1. Secure Water Treatment Dataset . . . 41
4.2. Proposed Data-based Curriculum Learning for Anomaly Detection . . . 43
4.2.1. Curriculum Design Based on Sequence Length . . . 44
4.2.2. Curriculum Design Based on Window Size . . . 45
4.2.3. Curriculum Design Based on Time Series Stationarity . . . 46
4.2.3.1. Augmented Dickey-Fuller Test . . . 46
4.2.4. Curriculum Design Based on Data Noise Level . . . 50
4.3. Proposed Model-based Curriculum Learning for Anomaly Detection . . . 52
4.4. Proposed Data-based and Model-based Curriculum Learning for Anomaly Detection . . . 54
4.5. Experiments . . . 55
4.5.1. Data Preprocessing . . . 56
4.5.2. Model Architecture . . . 57
4.5.3. Training Scheduler Design . . . 59
4.5.4. Anomaly Detection Results and Discussion . . . 61
4.5.5. Comparison with State-of-the-Art Models . . . 64
4.6. Summary of the Chapter . . . 65

5. ANOMALY DETECTION IN TEXTILE PROCESSES 66
5.1. Textile Processes 66
5.1.1. Control of Dyeing Processes 67
5.1.2. Challenges in Anomaly Detection for Textile Processes 69
5.2. Textile Dyebath Temperature Dataset 70
5.3. Experiments 71
5.3.1. Data Preprocessing 72
5.3.2. Anomaly Detection Results and Discussion 74
5.4. Summary of the Chapter 77
6. CONCLUSION 78
REFERENCES 79

LIST OF FIGURES

Figure 1.1. Examples of a point anomaly, contextual anomaly, and collective anomaly. 3

Figure 1.2. General anomaly detection schematic diagram. 4

Figure 3.1. A multilayer perceptron network. 24

Figure 3.2. Activation functions in artificial neural networks. 26

Figure 3.3. An example of 2D convolution as performed in convolutional neural networks. 27

Figure 3.4. Internal structure of the long short-term memory cell. 29

Figure 3.5. General structure of an autoencoder. 31

Figure 3.6. MSCRED framework: (a) Signature matrices encoding (b) Temporal pattern modeling (c) Signature matrices decoding (d) Anomaly detection. 32

Figure 3.7. Confusion matrix for anomaly detection evaluation. 40

Figure 4.1. Hierarchical control scheme. 42

Figure 4.2. Examples of time series data from the downsampled SWaT test set and labeled anomaly locations. 42

Figure 4.3. Proposed data-based curriculum learning for multivariate time series anomaly detection framework. 43

Figure 4.4. Data-based curriculum learning for multivariate time series anomaly detection. 44

Figure 4.5. Some of the more stationary time series data of the downsampled SWaT training set. 49

Figure 4.6. Some of the less stationary time series data of the downsampled SWaT training set. 49

Figure 4.7. Easy, medium, and hard time series data of the AIT-203 sensor of the downsampled SWaT training set. 51

Figure 4.8. Proposed model-based curriculum learning for multivariate time series anomaly detection framework. 53

Figure 4.9. Model-based curriculum learning for multivariate time series anomaly detection. 54

Figure 4.10. Reconstruction-based multivariate time series anomaly detection. . 55

Figure 4.11. F1 scores of the MSCRED model at various learning rates and epochs using Adam optimizer. 59

Figure 4.12. (Left) Training scheduler with a total of 15 epochs. (Right) Training scheduler with 15 epochs per training round. 60

Figure 5.1. Original training set of the textile dyebath temperature dataset. . 70

Figure 5.2. Cleaned training set of the textile dyebath temperature dataset. . 72

Figure 5.3. Test set of the textile dyebath temperature dataset with an oscillation anomaly. 73

Figure 5.4. Test set of the textile dyebath temperature dataset with noisy temperature readings. 73

Figure 5.5. Test set of the textile dyebath temperature dataset with an oscillation anomaly and noisy temperature readings. 74

Figure 5.6. Collective anomaly detection on a test set with an oscillation anomaly at various reconstruction error thresholds. 75

Figure 5.7. Collective anomaly detection on a test set with noisy temperature readings at various reconstruction error thresholds. 75

Figure 5.8. Collective anomaly detection on a test set with an oscillation anomaly and noisy temperature readings at various reconstruction error thresholds. 76

LIST OF TABLES

Table 4.1. Anomaly detection results on SWaT dataset using MSCRED at various learning rates and epochs. 58

Table 4.2. Anomaly detection results on SWaT dataset with two difficulty levels using different training schedulers. * represents training with easy subset for 5 epochs and then with whole training set for 10 epochs. ** represents training for 15 epochs at each training round. 60

Table 4.3. Anomaly detection results on SWaT dataset with three difficulty levels using different training schedulers. * represents training with easy subset for 5 epochs at each training round. ** represents training for 15 epochs at each training round. 61

Table 4.4. Anomaly detection results of the DCL-MSCRED-W3 model on SWaT dataset at different reconstruction error threshold values. 61

Table 4.5. Anomaly detection results of designed curriculum learning-based methods on SWaT dataset. 62

Table 4.6. Anomaly detection results of the best curriculum learning-based design and the state-of-the-art techniques on SWaT dataset. 64

Table 5.1. Description of the test sets of the textile dyebath temperature dataset. 71

LIST OF SYMBOLS

A	Anomaly score
C	Difficulty measurer
h	Hidden units of multilayer perceptrons
I	Image matrix
K	Kernel matrix
$\mathcal{L}$	Loss function
$\mathbf{M}^t$	Correlation matrices
n	Data dimension
N	Time series length
t	Time step
x	Single data point
$\mathbf{x}$	Time series data
$\mathbf{X}$	Matrix
θ_A	Anomaly threshold
θ_R	Reconstruction error threshold
Θ	Deep learning model
σ^2	Variance
ω	Window size

LIST OF ACRONYMS/ABBREVIATIONS

1D	One Dimensional
2D	Two Dimensional
ADF	Augmented Dickey-Fuller
AI	Artificial Intelligence
ANN	Artificial Neural Network
AutoML	Automated Machine Learning
CIM	Computer Integrated Manufacturing
CL	Curriculum Learning
CNN	Convolutional Neural Network
DCL	Data-based Curriculum Learning
FSM	Finite State Machine
ICA	Independent Component Analysis
ICS	Industrial Control System
LSTM	Long Short-Term Memory
LSTM-ED	Long Short-Term Memory Encoder-Decoder
ML	Machine Learning
MLP	Multilayer Perceptron
MSCRED	Multi-Scale Convolutional Recurrent Encoder-Decoder
MVTS	Multivariate Time Series
NASA	National Aeronautics and Space Administration
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
RNN	Recurrent Neural Network
SCADA	Supervisory Control and Data Acquisition
SELU	Scaled Exponential Linear Unit
SPL	Self-Paced Learning
SSM	System Signature Matrices

1. INTRODUCTION

In industries, as machines have become interconnected, anomaly detection has become more important to ensure high-quality products and safety to the surroundings. In this thesis, a comprehensive study of anomaly detection in industrial processes is performed, particularly in water treatment [1] and textile processes [2]. This chapter outlines anomaly detection, the motivation behind this study, contributions, and organization of the thesis.

In everyday life, we encounter various situations that deviate from the standard norm and we are able to identify such occurrences using our natural intelligence. Similarly, the industry can encounter such instances in which a data point or set of data points can significantly deviate from the rest of the sample distribution. Such an occurrence is known as an *anomaly* and the process of identifying these data points is referred to as *anomaly detection*. The term anomaly is used to describe various situations across different fields in the industry. The variable use of this term ranges from malicious software-related behavior in network security to defective products in the manufacturing industry. To prevent serious problems like industrial control system failure, network intrusion, or financial fraud, it is essential to detect anomalous behaviors at their early stages.

In the past, data generated by industrial sensors and actuators was typically small in dimension and mainly limited to single variables. A sensor is a device that converts a physical quantity into an electronic output whereas an actuator is a device that converts a signal into a physical output such as turning a water pump on or off. The data that consists of only a single variable collected over a period of time is known as univariate time series. Modern cyber-physical systems generate large amounts of high-dimensional data through sensors from interconnected machines. This data which contains multiple variables is known as *multivariate time series* (MVTS) [3]. Due to the complex temporal correlations within these variables, identifying anomalies in

industrial processes is a very difficult task.

The control industry has seen an unprecedented advancement in recent years, with a major shift from manual to automatic techniques owing to computer integrated manufacturing (CIM). CIM implementations make use of cutting-edge computer hardware, networks, and sensors to create a standard database, collect pertinent information, and deliver it to the user interface [4]. Due to the variety and complexity of automation systems, communication channels, and proprietary protocols in CIM, anomaly detection in industrial processes is complicated and, thus, is an area of research that needs significant attention.

Industrial processes are monitored by several industrial control systems (ICS) which may consist of proprietary protocols. ICS are electronic systems that carry out control functions and maintain the safety of several types of industrial infrastructure. A proprietary protocol is a communications system that is privately possessed by a company or person and, therefore, in multiple instances, does not contain sufficient technical information and detailed specifications. This is because industrial networks frequently include components from specialized manufacturers, each of which uses non-standardized technology [5].

1.1. Multivariate Time Series Anomaly Detection

Anomalous behavior in time series can be divided into three categories: *point anomalies* which are singular data points with significant statistical deviation from the rest of the data points, *contextual anomalies* which are points that are within the standard range of the time series but show unexpected temporal behavior and can only be detected under specific conditions, and *collective anomalies* which are a group of statistical outliers but each data point may be classified as normal when considered separately [6]. Examples of these anomalies are shown in Figure 1.1.

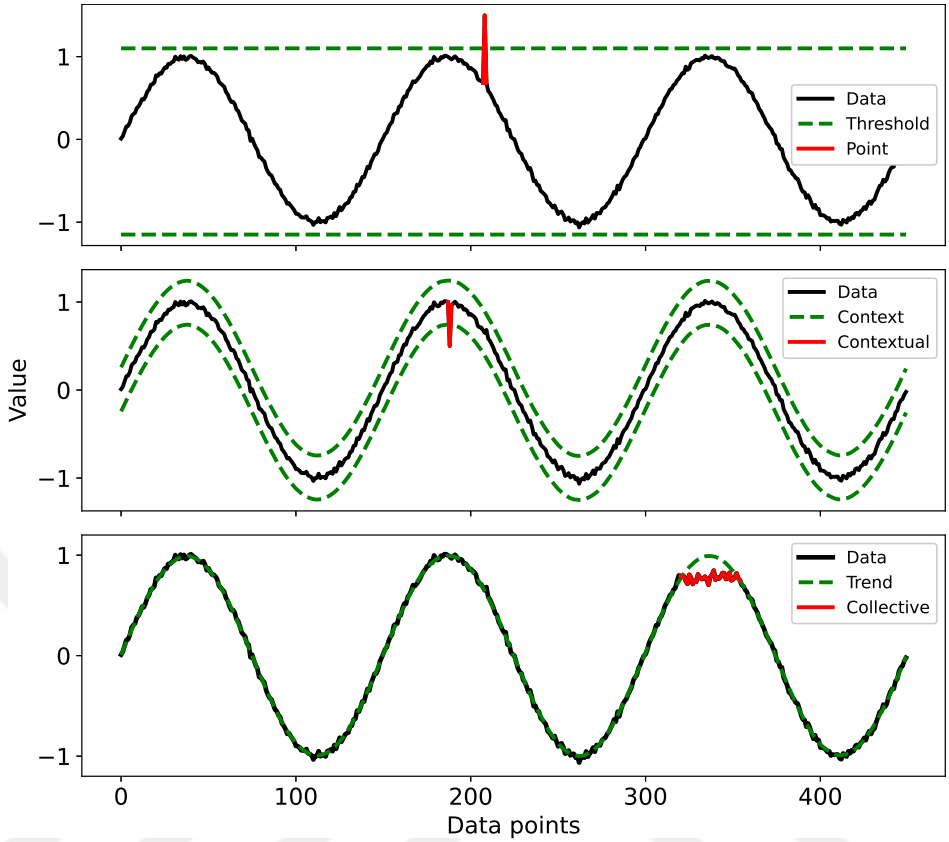

Figure 1.1. Examples of a point anomaly, contextual anomaly, and collective anomaly.

The concept of anomaly detection is simply based on differentiating anomalous system behavior from the standard or expected activity [7]. For instance, anomaly detection on temperature data is based on identifying anomalous data points from the temperature profile by comparing the process profile with the theoretical or expected graph. An anomaly detection framework should be capable of learning normal temporal patterns during training and identifying any novel pattern during testing as an anomaly. Temporal patterns in time series analysis are systematic or recurrent variations in the data. They depict trends or regularities that recur or adhere to a particular pattern at various points in time within the data [8]. Identifying and understanding these patterns is essential for anomaly detection. The following are examples of how temporal patterns can appear:

- Trend: This is a long-term pattern of data movement or direction. It displays whether the data is increasing, decreasing, or staying mostly unchanged over a

long window.

- Seasonality: This describes the regular periodic fluctuations in the data. The frequency of these cycles might be daily, weekly, monthly, or seasonally, based on the type of data and the occurrence under observation.
- Cyclic Patterns: These are more unpredictable and irregular compared to seasonality, which has fixed and predictable periods. Cycles mainly occur across longer periods.
- Autocorrelation: This is the relationship between a data point and previous data points in the same time series. When data points have positive autocorrelation, they tend to be similar to the previous points whereas negative autocorrelation shows that they are distinct.

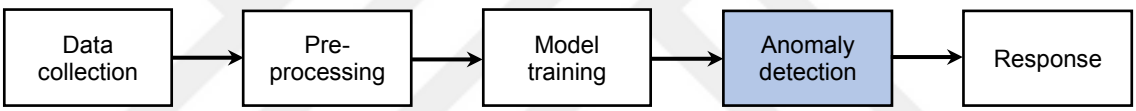

Figure 1.2. General anomaly detection schematic diagram.

Given a MVTS of dimension n and length N, denoted as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n) \in \mathbb{R}^{N \times n}$, where $\mathbf{x}_i \in \mathbb{R}^N$. For anomaly detection, generally, the algorithm outputs a vector $\mathbf{y} = \{0, 1\} \in \mathbb{R}^N$, where "0" indicates normal data points and "1" indicates anomalies. Figure 1.2 shows a generic block diagram of anomaly detection, starting with collection of streaming data from the sensors. The data is then cleaned in the preprocessing stage, followed by training an anomaly detection model. Subsequently, anomalies are identified and necessary action is taken to mitigate their effect.

Anomaly detection is usually applied in the data preprocessing stage in several fields which include cybersecurity, sensor networks, system health monitoring, machine vision, and fraud detection among others. The adoption of internet-of-things in the industry has increased their vulnerabilities to cyberattacks. Therefore, it is crucial for the society to protect vital infrastructures because terrorists and hackers deem industrial infrastructure like power plants and water supply networks to be important

targets when they are seeking attention. Anomaly detection has been a prominent research area in the history of ICS and has even gained more attention recently, owing to the advancement of adaptive algorithms. The development of security for internet-of-things has inspired the design of innovative techniques [9].

Generally, anomaly detection techniques can be divided into three major categories: *supervised*, *semi-supervised*, and *unsupervised* [10]. These three categories differ in the nature of the training data used to design the models as explained below:

- Supervised anomaly detection makes use of training datasets with both labeled normal and anomalous data points. The underlying premise of this technique is to make a comparison between the model created from normal data and the one created from anomalous data. The main drawback of this approach is the difficulty in finding exact or appropriate labels for the data.
- Semi-supervised anomaly detection makes use of only normal data points in the training process. Hence, any data point that cannot be categorized as normal is thus labeled as an anomaly. Such an approach is useful in real-world applications in which some of the datasets are partially labeled. Since semi-supervised approaches assume that the training data belongs only to normal points, they are preferable to supervised techniques.
- Unsupervised anomaly detection approaches assume that the training datasets are unlabeled. Furthermore, it assumes that there are more normal points than anomalies in the test datasets. However, if such a hypothesis is incorrect, the strategy may result in high false alarm rates.

1.2. Motivation of the Thesis

Researchers have developed advanced techniques, nonetheless, several drawbacks still need to be addressed. Anomaly detection methods that rely on simple user-defined rules cannot adequately handle complex systems. Moreover, the training of traditional methods is a lengthy process before the models can be deployed. This proves

disadvantageous as such delays could result in failure to identify an anomaly during system operation. Furthermore, in anomaly detection, fully labeled data is rarely available, hence, rather than using supervised algorithms, it is beneficial to employ techniques that make use of semi-supervised or unsupervised learning algorithms. In order to identify anomalies, it is often assumed that a significant fraction of the data indicates normal behavior. Anomaly detection algorithms can be trained using this data and the resulting model can be used in real-life situations to identify anomalous patterns [6].

The advent and successive increase in industrial process automation and the interconnectedness of the equipment and control centers have dramatically introduced novel vulnerabilities to the control industry. For example, a water treatment plant is made up of several unique hardware and software components designed by different companies. A number of these devices operate according to customized non-standard protocols. Consequently, the experts who operate these components are often unaware of the specifications of the systems they assemble or operate on a daily basis. As such, it is exceedingly challenging to employ conventional anomaly detection methods for ICS. Anomalies in several ICS have increased and some of the examples are as follows:

- In August 2022, the National Aeronautics and Space Administration (NASA), United States, postponed their planned launch of the space launch system, a new Moon rocket which is part of their Artemis program. The space launch system is the largest rocket that NASA has ever engineered and is aimed at carrying the astronauts for the moon expedition. The power unit must be gradually cooled down to the proper operating temperature before liftoff, however, the controllers of the rocket's engine had difficulty in doing so, leading to the halt [11].
- In May 2021, Colonial Pipeline company, a United States firm which has significant control in the gasoline pipeline industry had to suspend operations due to a cyberattack which resulted in fuel shortage and price increase. The hackers stole a significant quantity of data and encrypted computer systems with ransomware. Eventually, the systems were restored back with help from a decryption soft-

ware [12].

- In February 2021, hackers remotely gained access to the computer networks of a water treatment facility in a small city in Florida, United States, and attempted to poison the water supply. They increased sodium hydroxide to unsafe levels which might have caused serious harm to the locals. The intrusion was alleviated before contaminating drinking water [13].
- In December 2017, hackers breached crucial safety systems of a power plant in Saudi Arabia and they remotely controlled the systems using a complex malware named Triton. The factory was able to detect the intrusion because several controllers switched into a fail-safe mode when the intruders tried reprogramming them [14].
- In January 2016, Google's Nest learning thermostat encountered a software bug that left customers in the United States unable to adjust the temperature in their houses. The software fault, which coincided with low temperatures caused the batteries of the device to self-discharge. This problem was brought on by a software upgrade as well as problems resulting from the use of outdated air filters and incompatible boilers. Eventually, the problem was solved for the majority of the affected clients [15].
- In 2013, several Boeing 787 airplanes experienced electrical system faults resulting from lithium-ion batteries. This Dreamliner had encountered fuel leaks, broken cockpit glass, braking problems, and electrical fires prior but the faults emanating from the batteries such as overheating, were the main point of concern. Citing safety fears, all 50 of the 787 Dreamliners worldwide had to be temporarily withdrawn from operation [16].
- In August 2012, Saudi Aramco in Saudi Arabia, one of the world's largest petroleum and natural gas company was attacked by a self-replicating malicious software that affected their desktop computers. The attack did not affect the plant machinery, computer control systems, and production process. However, it took them about two weeks to get the systems back to normal operation [17].

The integration of the ICS with anomaly detection frameworks can protect the systems from attacks or minimize the extent of damage to the equipment. These techniques can identify anomalies in the machinery during the manufacturing process and prevent the deployment of faulty equipment. This can enhance the overall efficiency of the manufacturing process and ultimately result in high-quality products. Moreover, identifying and addressing such problems can significantly reduce expenses that may be incurred from repairs or replacement of anomalous systems. As seen from the given examples, anomalies are common in potentially dangerous environments which may even lead to loss of lives. Anomaly detection techniques can identify safety problems resulting from faulty equipment or human errors, thus, paving the way for problem correction before any accidents occur. In addition, the use of these techniques can minimize unexpected interruptions or equipment failures which results in increased customer satisfaction.

Anomalous behaviors in ICS must be precisely identified and immediately alleviated since they can pose serious harm to personnel and property. Water treatment plants and textile dyehouses have several machines that work individually and simultaneously. Also, a centralized computer constantly communicates with these machines for data transfer. Anomalies like pay overload, abnormal bandwidth usage, link failure, or suspicious device behavior can occur while the machines operate. For instance, in textile processes, some of these anomalies can cause the temperature profile of the dyeing process to have a significant deviation from the expected graph. By constantly monitoring the temperature during the industrial processes, such anomalies can be detected and mitigated before further damage occurs.

1.3. Contributions of the Thesis

Anomaly detection has been studied across a variety of industries with the focus mainly being designing high-capacity frameworks. Since many proposed frameworks in MVTS anomaly detection are based on deep learning, how the model is trained can have a tremendous effect on the model performance. Bengio *et al.* [18] introduced

curriculum learning (CL), which is a strategy that involves training a model on data samples progressing from "easy" to "difficult". In the literature, CL has proven to be capable of mitigating catastrophic forgetting, enhancing convergence, and improving model performance in some applications such as in federated face recognition [19] and natural language understanding [20]. Despite these advantages, prior research by Wang *et al.* [21] shows that the benefits of CL cannot be generalized, thus, necessitating the need of analyzing domain-specific effects of CL-based designs.

While existing literature has studied the efficacy of CL in several areas, its application to MVTS anomaly detection remains an unexplored area. Inspired by Koenecke and Gajewar [22] who proposed a curriculum based on the forecast horizon for time series forecasting, this thesis aims to study a wide range of time series features that can be used as curricula. In MVTS anomaly detection, to the best knowledge of the author, no research has performed an extensive study of the benefits and limitations of CL. The main contributions of this thesis can be summarized as follows:

- Introduces CL to MVTS anomaly detection and conducts several experiments using a semi-supervised deep learning model to analyze the performance of CL-based designs.
- Proposes data-based CL approaches for MVTS anomaly detection and further determines the most important data feature for CL-based MVTS anomaly detection.
- Studies whether a curriculum based on the model complexity can outperform the data-based techniques. Moreover, these two curricula are combined to analyze the effect of each approach.
- Introduces the concept of representing data as signature matrices for anomaly detection in textile processes, an area that has received limited attention.

1.4. Organization of the Thesis

The rest of the thesis is structured as follows: Chapter 2 provides a study of some of the anomaly detection techniques including simple probabilistic and statistical models and advanced deep learning algorithms. Subsequently, Chapter 3 outlines preliminary information on the deep learning methodology, gives an overview of the model used in this thesis, and provides detailed information on how to design a curriculum, and the evaluation metrics. Chapter 4 presents the proposed frameworks and performs MVTS anomaly detection on a benchmark dataset. It then compares the proposed approach to the state-of-the-art techniques. In Chapter 5, a background of some of the process control techniques in the textile industry is provided. This chapter also explains how textile data was collected, cleaned, and preprocessed. Then, how anomaly detection in textile processes is conducted is explained. Chapter 6 summarizes the proposed techniques, main outcomes of this research, and outlines the possible future direction in this research area.

2. OVERVIEW OF ANOMALY DETECTION TECHNIQUES

In MVTS anomaly detection, learning temporal patterns is crucial to understanding the underlying structure of the normal behavior of a system. However, this is a difficult problem since MVTS inherently have complicated patterns. Traditional approaches use simple conditional rules to generate an alert in case a fault occurs in the system. However, such an approach needs a priori threshold value of the target variable. In practice, the manufacturer ought to provide or make an estimate of this value. The threshold value may change over time if the system includes non-stationary parameters. Due to the non-monotonic nature of MVTS data, such conditional rules-based approaches are not applicable. This necessitates the development and evaluation of several techniques as presented in this chapter.

2.1. Probabilistic and Statistical Models

In probabilistic and statistical modeling, the data is assumed to follow a certain probability distribution. The model parameters are then learned from the selected distribution. Some of the modeling choices are Bernoulli and Gaussian probability distributions. Bernoulli distribution is a discrete distribution in which the outcomes are classified into "success" and "failure" with probabilities of p and $1 - p$, respectively. Therefore, this distribution can be chosen to model categorical data. Gaussian probability distribution is a continuous distribution and therefore, can be selected for real-valued data. In this distribution, the model parameters can be learned by utilizing the expectation-maximization method. From this approach, the data points are put into different clusters and the quality of the density-based fit is obtained. Then from these outcomes, the data points with low fit values may be regarded as anomalies [23].

In statistical modeling, extreme values are usually assumed to be anomalies. Furthermore, some statistical tests may be applied to identify these anomalies. The main

benefit of probabilistic models is that they offer a general expectation-maximization-based approach that can be easily applied to any particular form of data which is usually not the case in other modeling techniques. On the other hand, probabilistic models may not produce a good fit if an inappropriate distribution is chosen. A further disadvantage is that they are most likely to overfit when trying to model significantly large data [23]. Kromkowski *et al.* [24] introduced a seasonal autoregressive integrated moving average anomaly detection model. Such a technique demands considerable training time and may fail to effectively perform in high-dimensional MVTS.

Grzesiek *et al.* [25] proposed a statistical testing methodology for long-term temperature data processing for anomaly detection. They proposed the application of Student's T-statistic and Wilcoxon test to some data moments such as mean and median rather than the raw data to accurately determine the structure break point in the temperature data. Then they used the data collected from a commercial supervisory control and data acquisition (SCADA) system which contains the temperature data of a belt conveyor system gearbox of a deep underground mine for evaluation of their approach. In conclusion, they claim that the suggested technique is accurate for the problem under consideration as the observed structure breakpoint matched the period of repair obtained from the maintenance team.

2.2. Finite State Machine Models

Finite state machines (FSMs) are mathematical models that consist of a set of states and transitions that are executed when specific conditions are satisfied [26, 27]. The FSM technique is used in modeling and designing synchronous digital circuits. FSMs play a vital role in the design of complicated digital systems. The states keep a record of the previous input such that a transition is performed in response to the next input and transition conditions. A physical realization of a system with past data that cannot be represented in the form of a discrete number of states is impossible. The process of developing FSMs is straightforward and the result is easy to explain.

There are two classes of digital circuits namely sequential and combinational. The output vector of a sequential circuit relies on prior system states which implies that they have memory. On the other hand, the output vector of a combinational circuit simply relies on the present input vector, thus, they are memoryless. The transition from one state to another occurs if the respective transition conditions are satisfied and are typically reviewed at every clock edge, hence, the operation is synchronous. FSMs offer a mechanism for developing sequential circuits that may produce solutions that are very close to optimal or even optimal. In addition, this technique can function without a priori specifications of the expected design of the general circuit of the problem under consideration. It is typically more difficult to design and implement accurate state machines in hardware than in software.

State diagrams and tables, which are analogous techniques of evaluating all transitions are typically used to model FSMs. A state diagram is a visual depiction of the features and specifications of a synchronous circuit. It must show all potential system states, conditional state transition must be executed only if the conditions are fulfilled, and for standard architectural implementation, the output vectors have to be precisely identical in every state. There are two types of state machines which are Moore and Mealy. A Moore state machine is a FSM in which the input has no direct effect on the output. The output only relies on the present state of the machine and it only changes with respect to the clock, hence, the circuit is synchronous. A Mealy state machine is a FSM in which the input has a direct effect on the output. The output relies on both the present state and input state of the machine. It can change regardless of the clock, hence, the circuit is not synchronous [26, 27].

FSMs can be easily visualized using state diagrams to facilitate the analysis and interpretation of the results. The correlation and clusters within the data points of any particular industrial process can be uncovered through data analysis and then FSMs may be used to identify anomalies. The first step in this approach is to construct a FSM that represents the expected behavior of the system by analyzing the data and identifying the fundamental states and transitions. Then anomaly detection can be performed

by applying the FSM to the data and comparing the observed transitions in the data to those in the FSM. If the observed transitions deviate from the transitions in the FSM, there may be an anomaly in the data. The probability of the observed transitions may be used as the evaluation metric to identify the most significant anomalies.

An advantage of FSMs is that they are capable of capturing the essential features of a process without the need for extensive data preprocessing. On the other hand, FSMs can be sensitive to the data distribution. Consequently, they may not be able to accurately capture the normal behavior of the system leading to high false positives or false negatives in the results. To address this challenge, researchers have developed a number of alternative approaches such as stochastic FSMs and adaptive which are capable of adapting FSMs to the data. These techniques can enhance the robustness of FSMs to the data which may in turn improve the performance of the algorithm. A further challenge of using FSMs for anomaly detection is that they can be computationally intensive.

Cheng *et al.* [28] presented a security approach named Orpheus to protect cyber-physical control systems from data-oriented cyberattacks. They further introduced an event-aware FSM, a model that inspects the events occurring in the control systems to identify anomalies and data-oriented attacks. Orpheus framework is the application of event identification to already existing models such as n-grams, FSM, and hidden Markov models to improve their anomaly detection efficiency. Real-world analysis showed that their proposed strategy is feasible and effective. Nevertheless, due to the variety of cyber-physical hardware, the horizon of evaluation of their approach is limited by a lack of benchmark systems. Moreover, an anomaly that does not result in a discrepancy between the control structure and physical environment may not be detected.

2.3. Machine Learning Models

The utilization of machine learning (ML) algorithms for anomaly detection has significantly expanded in recent history. ML, a sub-field of artificial intelligence (AI) built upon natural human intelligence, uses data and algorithms to progressively learn and improve predictive performance without being explicitly programmed. The goal of ML is to automate the learning process by training the models with relevant historical data. The trained ML model can classify normal data points from anomalous ones and factors such as the choice of the algorithm, quality and quantity of the data affect the accuracy of the model. There exist several ML techniques, this section presents linear, proximity-based, and automated ML (AutoML).

2.3.1. Linear Models

Anomaly detection and prediction concepts are mainly based on correlation analysis, a statistical evaluation method that measures the strength of the linear relationship between two variables in the data. Since anomalies are outcomes that differ significantly from the predictions obtained from a prediction model, linear modeling is a viable option for identifying such anomalous points. Linear models assume that the normal data points lie within the lower-dimensional subspace.

Regression analysis, a parametric correlation analysis considers the dependent and independent variables and aims to determine the relationship between them. For instance, regression may be used to find the 1-dimensional (1D) line that best fits the data represented in a 2-dimensional (2D) space. The ideal lower-dimensional hyperplane is typically found using the least-squares method. In the scope of anomaly detection, the magnitude of the deviations of the data points from the determined hyperplane can be utilized as the anomaly scores. A threshold can be chosen such that the anomaly detection rate increases without excessively increasing the false alarm rate. Then, using extreme-value analysis, it is possible to determine the set of points with significantly large deviations [23].

Principal component analysis (PCA), a non-parametric correlation analysis method based on dimensionality reduction can also be used in anomaly detection. PCA offers the lowest reconstruction error after projecting to the lower-dimensional subspace in which normal data points are embedded. Since anomalous points do not lie in this subspace, they exhibit high reconstruction errors. As such, reconstruction error can utilized as the anomaly score in this technique. An advantage of this approach is that it can be applied even to incomplete datasets [23].

Zhu *et al.* [29] proposed a Temperature-driven Moving PCA (Td-MPCA) approach for structural anomaly detection. In order to improve the efficiency of moving PCA (MPCA), Td-MPCA includes the concept of blind source separation for thermal identification. First, fast independent component analysis (ICA), a method based on maximizing non-Gaussianity, is used to identify and analyze structural strains due to temperature. Then MPCA is used to identify anomalies in the separated temperature-related response. The operation of Td-MPCA can be classified into pre-indication, blind separation, and anomaly detection. To determine the total number of intrinsic components, the PCA is first performed on all target strain data. Then the minimum essential number of measurements for the input channels will be chosen for ICA separation. Next, ICA is applied to each of these different combinations separately. Subsequently, temperature-related strain is chosen among ICA estimators for each collection based on its correlation with local temperature data. Lastly, the strain with the strongest correlation with temperature variation is recorded and applied to MPCA for anomaly detection.

2.3.2. Proximity-based Models

Proximity-based methods are among the commonly used approaches for anomaly detection. These techniques involve identifying data points that are statistical outliers in the dataset based on similarity or distance metrics. The three main classes of proximity-based models are clustering, density-based, and nearest-neighbor methods. Clustering methods identify anomalies by comparing data points to groups or clusters of

similar points. These techniques assume that most of the data points belong to a cluster whereas anomalous points are isolated. Density-based approaches identify anomalies by comparing the density of data points in a given area. These approaches assume that the majority of data points are concentrated in certain areas while anomalous points are in less dense regions. Nearest-neighbor, also referred to as distance-based method, identifies anomalies by considering the distance of each data point from a specified number of nearest neighbors. In this technique, it is assumed that most of the normal data points are relatively close to each other while anomalies are far from the rest.

The advantage of proximity-based methods is that they are generally easy to understand and implement. They do not require a lot of prior information or assumptions about the data and can be applied to a wide range of datasets. However, determining the appropriate similarity or distance metric is a main challenge in the implementation of these techniques. Different metrics may produce inconsistent results. Another limitation of proximity-based methods is that they may not be able to identify anomalies if all the data points in a dataset do not form a cluster or are not concentrated in certain areas. Nevertheless, proximity-based methods can be effective for anomaly detection in a wide array of applications [23].

Li *et al.* [30] proposed a data-driven unsupervised shape clustering approach based on evaluating the similarity of the thermal readings to identify anomalies in batteries. The readings are constantly sorted into several groups and anomalies are identified from the deviations in the clusters. First, the data is divided into small segments for offline implementation which are then clustered appropriately. The data is continually obtained and buffered as a segment for online implementation before being processed. Secondly, invalid readings such as those beyond the operating range of the sensor are discarded. Next, data is standardized and filtered by excluding segments that do not exhibit any evident changes. Then the k-shape clustering algorithm is used to form clusters and lastly, the identified anomalies are confirmed. They claim that this technique is more resilient to data loss and can be implemented with little reference data compared to model-based or other data-driven approaches. Furthermore, this

approach has high accuracy and can give early warnings of unanticipated problems.

2.3.3. Automated Machine Learning Models

Most commercially available anomaly detection techniques prove highly efficient, however, they are difficult to employ in big companies that work with vast amounts of data. For an anomaly detection algorithm to be effective, its parameters must be precisely chosen because automatic tuning requires labeled data which is usually unavailable, thus, supervised anomaly detection algorithms cannot be applied. On the other hand, manual tuning on large data is practically impossible. Moreover, there is no general method that performs consistently well across different forms of data. Research on the application of AutoML methods in anomaly detection to counter such drawbacks is yet to receive significant attention. The fundamental idea behind AutoML is to automatically tune the model hyperparameters to maximize performance. Large organizations often work with different time series data, hence, it is a demanding task to choose from a large pool of algorithms which one to employ. In addition, it is challenging to set up an evaluation metric for anomaly detection without ground truth data [31, 32].

Chatterjee *et al.* [32] introduced an AutoML-based model selection and parameter tuning framework for application in anomaly detection. Without a specialized understanding of anomaly detection techniques or time-consuming manual labeling, their proposed technique enables big companies to design specific algorithms for various time series data. In their framework, they start by using a generative model to imitate the properties of the unlabeled data and then inject distinct types of anomalies at predetermined points to generate labeled training data. Then they apply a supervised learning algorithm on this data to choose the optimal parameters and most effective anomaly detectors. They compare and contrast their framework with a wide range of time series anomaly detection techniques available in their kit to analyze time series data. Results show that using AutoML constantly outperforms any specific approach using the F1 score as the evaluation metric.

2.4. Deep Learning Models

MVTS data consists of periodicity and seasonality components which make traditional anomaly detection approaches such as proximity-based methods incapable of detecting anomalies in streaming data with high accuracy. Deep learning has emerged as an effective method for extracting important features from large MVTS data, as evidenced by studies presented by Wagner *et al.* [33]. A lot of training data is generally needed to guarantee high standard performance of deep learning-based techniques. They can be classified into three main categories: reconstruction-based, forecasting-based, and generative models.

2.4.1. Reconstruction-based Models

Reconstruction-based methods typically involve utilizing an encoder to capture spatial information from normal data, followed by a decoder to reconstruct the data, with the reconstruction errors being used as anomaly scores. The underlying principle is that the model should effectively learn only normal patterns and poorly reconstruct previously unseen patterns so that they are tagged as anomalies.

Thill *et al.* [6] introduced a novel unsupervised learning, dilated convolution-based temporal convolutional network autoencoder (TCN-AE). TCN-AE is made up of an encoder and a decoder which are concurrently trained to discover a compact description of the input data and reconstruct the original input respectively. Then the reconstruction error is used to identify anomalous events. Since TCN-AE is a convolutional framework, it has the typical advantages of convolutional networks which include weight sharing, translation invariance, and computational effectiveness. As such, they are particularly ideal for computer vision applications and time series processing in which 1D convolutions are generally utilized. Unlike standard autoencoders, TCN-AE uses dilated 1D-convolutional layers rather than dense layers. As a result, this architecture is more adaptable when dealing with inputs of different sizes and can consider temporal correlations in the data with great ease.

The performance of the algorithm was evaluated using comprehensive real-world electrocardiogram measurements of 25 patients with cardiac arrhythmia. The F1 score of TCN-AE was compared with the F1 scores of ten different unsupervised anomaly detection algorithms. First, it was observed that the variants of TCN-AE perform remarkably better than the rest of the benchmark algorithms. A further notable observation was that TCN-AE performed the anomaly detection task five times faster than the runner-up algorithm. Overall, deep learning techniques outperformed non-deep learning approaches on the electrocardiogram readings. The significant learning improvement was attributed to the skip connections in the network. Additionally, the presence of hidden representations in the TCN-AE architecture is notably important in the model performance. The results also show that temporal anomalies vary by time scales in time series data.

2.4.2. Forecasting-based Models

Forecasting-based approaches generally train on historical normal data, forecast future data, and then use the deviations between actual and forecast values as anomaly scores. Since time series data is a sequence of values, it may need to be transformed to a suitable format in the preprocessing stage depending on the choice of approach. For instance, it has to be converted to a grid of values when employing convolutional neural networks (CNNs).

A novel Deep learning-based Anomaly detection technique for Time series data (DeepAnT) was proposed by Munir *et al.* [34]. Since DeepAnT uses sophisticated deep learning architecture, it is capable of identifying a broad variety of anomalies including point anomalies, contextual anomalies, and discords, and is also effective when applied to non-streaming data. DeepAnT is made up of time series predictor and anomaly detector components. The time series predictor is used to forecast the next time stamps for a specified prediction window using deep CNNs. The forecast values are then tagged as normal or anomalous using the anomaly detector component. The performance of DeepAnT is, however, affected by the data quality. For instance, DeepAnT may try to

model anomalous data points as normal if the dataset contains a relatively high amount of anomalies. Another limitation of this approach is its susceptibility to adversarial attacks.

Gómez *et al.* [35] designed SUSAN, which is a deep learning-based method aimed at detecting anomalies in industrial systems, specifically those affecting sustainability such as carbon emissions and energy consumption. It consists of four modules: Data preprocessing module cleans the data and uses one-hot encoding to encode the data features; Sustainability-based features generation extracts the most important features for conducting anomaly detection; Model generation which selects the most suitable deep learning regression models, fine-tunes them via grid search, trains and validates them on the datasets; and the Sustainability anomaly detector uses the trained model to perform real-time anomaly detection in unseen data with the prediction errors as the anomaly scores. The main goal of the design of SUSAN is to maximize anomaly detection recall, even if it results in a reduction in precision.

2.4.3. Generative Models

Generative techniques employ a model to learn the distribution of the data and subsequently classify anomalies as data points with a probability below a specified marginal likelihood threshold. These methods generally employ generative models such as Gaussian mixture models, variational autoencoders, and generative adversarial networks.

Huong *et al.* [36] presented FedeX, a federated learning-based explainable anomaly detection for ICS for the purpose of identifying anomalies in a distributed setting. In this approach, computations are directly performed on decentralized devices. This circumvents the need for constant transfer of training data to the centralized server, thus, alleviating network latency problems [37]. FedeX is a hybrid method that uses a generative variational autoencoder, support vector data description, and explainable AI. By making use of explainable AI, features that are mostly responsible for the anomaly can

be identified. FedeX also makes it possible for systems to respond to anomalies more quickly as anomaly detection takes place close to their sources. Experimental results show that FedeX performs outstandingly in comparison to the considered strategies.

Lim *et al.* [38] proposed Multivariate anomaly detection with Score-based Generative Models (MadSGM), a technique that addresses the drawbacks of existing reconstruction-based, density-based, and gradient-based methods in determining the anomaly scores. The score-based generative models used by MadGSM are capable of generating high-quality samples and computing probability densities via the score functions of these samples. Its design pipeline consists of a conditional score network, a denoising score matching loss, and a purification process that improves the classification of anomalies and normal data. Such a design makes MadSGM robust and helps it to accurately detect a variety of anomalous patterns. The efficacy of MadSGM was evaluated through several experiments on five benchmark datasets and its performance was compared with nine benchmark approaches using the F1 score and the area under the curve as evaluation metrics. MadSGM outperforms other approaches which proves its robustness and accuracy in detecting anomalies in diverse datasets.

2.5. Summary of the Chapter

In this chapter, an overview of anomaly detection techniques, including a literature study of these techniques was presented. First, probabilistic and statistical modeling methods such as Bernoulli and Gaussian distributions were briefly explained. Then FSM modeling was studied by analyzing how FSMs can be used for anomaly detection, their advantages, and disadvantages. Next, some examples of ML models which are linear methods such as regression, proximity-based methods such as clustering, and AutoML are provided. Lastly, an outline of deep learning methods such as reconstruction-based, forecasting-based, and generative techniques is presented. To understand the fundamentals of these methods, Chapter 3 presents deep learning techniques used in this thesis. In Chapters 4 and 5, experiments using a reconstruction-based technique in water treatment and textile processes are conducted.

3. DEEP LEARNING METHODOLOGY AND EVALUATION METRICS

This chapter presents an overview of deep learning neural networks. An outline of the process of designing an anomaly detection model to learn from the data using different elements such as an optimization algorithm and a loss function is also presented. Moreover, evaluation metrics used to assess the model performance are provided.

3.1. Machine Learning Overview

ML is a part of AI that involves designing a computer algorithm that learns from the data to make predictions or decisions rather than explicitly programming it. According to Mitchell [39], "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." From this definition, learning is the method of acquiring the necessary ability to undertake the task. For instance, in this thesis, the learning process involves attaining the ability to perform the anomaly detection task.

ML can be utilized in a wide range of tasks, for example, in classification and regression. In classification tasks, the ML algorithm is required to generate a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$ for k categories in the dataset and then identify the class in which the given input represented as a vector $\mathbf{x}$ belongs to. The algorithm maps the input $\mathbf{x}$ to the class represented as a numeric code $\mathbf{y} = f(\mathbf{x})$. In these tasks, performance measures such as accuracy and F1 score can be used. In regression tasks, the ML algorithm is required to generate a function $f : \mathbb{R}^n \to \mathbb{R}$ that outputs a numerical value for a given input sequence of values. For these tasks, the performance measures are based on the error score between the actual and predicted values. Deep learning is a subset of ML that employs artificial neural networks (ANNs), also known as multilayer perceptrons (MLP), to learn from the data using an optimization algorithm [40].

3.2. Multilayer Perceptrons

Linear models such as linear regression or logistic regression have long been the basis of data analysis and ML because they are easy to understand and implement. However, when faced with complex, nonlinear correlations present in many real-world datasets, their efficacy decreases. By using one or more hidden layers, the limitations of linear models can be averted. One simple way to accomplish this is to stack many fully connected layers one after the other so that each layer transfers its output to the subsequent one until the final outputs are produced. Such a representation is referred to as MLP [40].

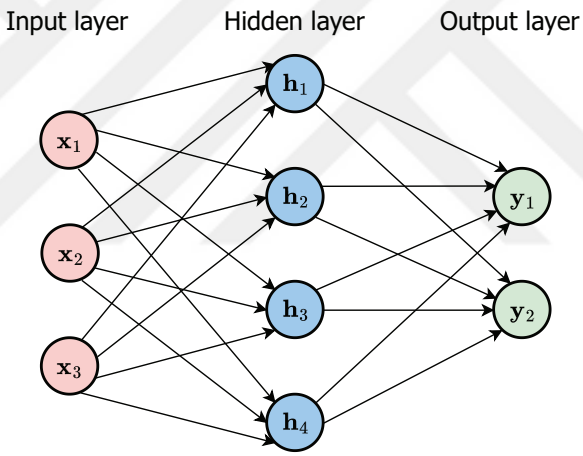

Figure 3.1. A multilayer perceptron network.

The MLP shown in Figure 3.1 has three inputs, four hidden layer neurons, and two outputs. Since there are no computations performed at the input layer to produce outputs from this network, calculations are performed at the hidden layers and output. Hence, this network is considered to be a two-layer MLP. It is important to note that these layers are fully connected, which means that each input affects every neuron in the hidden layer, which in turn affects every neuron in the output layer.

Let the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ represent a minibatch comprising of n data samples, each with d inputs (features), $\mathbf{H} \in \mathbb{R}^{n \times h}$ denote the hidden representations of an MLP with h hidden units. Given that both the hidden and output layers are fully connected,

the hidden layer weights are represented as $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$ with biases $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$, and the output layer weights as $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ with biases $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$. The hidden layer outputs $\mathbf{H}$ and network outputs $\mathbf{Y}$ are computed respectively as

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}, \tag{3.1}$$

$$\mathbf{Y} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}. \tag{3.2}$$

Upon adding the hidden layer, the model needs to update the additional parameters. The weights and biases are updated as $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)}$ and $\mathbf{b} = \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$, respectively. The MLP network output can be represented in a linear form as

$$\begin{aligned} \mathbf{Y} &= \left(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W} + \mathbf{b}. \end{aligned} \tag{3.3}$$

3.2.1. Activation Functions

To fully use the potential of multilayer architectures, nonlinear activation functions, σ are used at each hidden unit. The purpose of these functions is to introduce nonlinearity into the model, thus, enabling it to learn complex patterns and correlations within the data. The outputs of activation functions, $\sigma(x)$ are termed activations. Some of the commonly used activation functions are Rectified Linear Unit (ReLU), Sigmoid, hyperbolic tangent (tanh), and Scaled Exponential Linear Unit (SELU). They are shown in Figure 3.2 and can be computed respectively as

$$\mathrm{ReLU}(x) = \max(0, x), \tag{3.4}$$

$$\mathrm{Sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{3.5}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{3.6}$$

$$\mathrm{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot (e^{x} - 1) & \text{if } x \leq 0 \end{cases} \tag{3.7}$$

where λ and α are constants ($\lambda \approx 1.0507$ and $\alpha \approx 1.6733$).

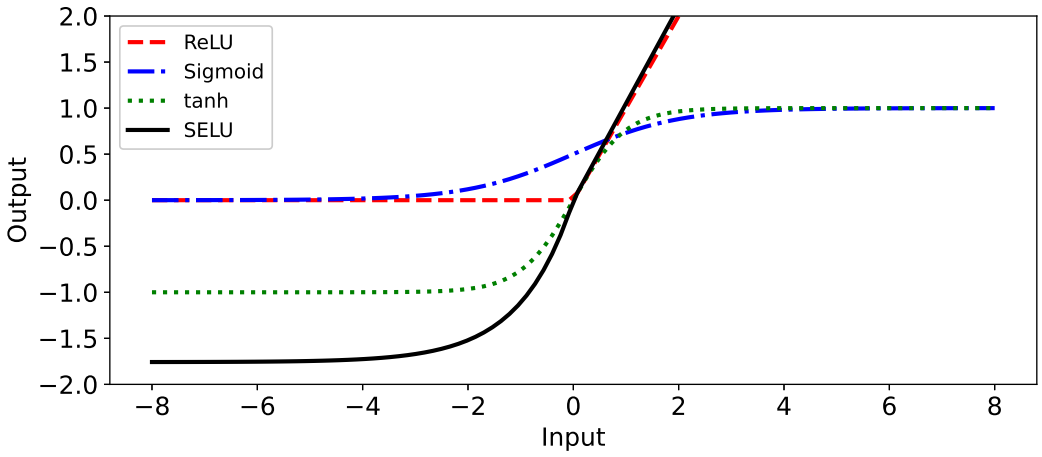

Figure 3.2. Activation functions in artificial neural networks.

By introducing the activation functions to the MLP, it is no longer possible to represent the network in a linear form. The hidden layer outputs and network outputs are now computed respectively as

$$\mathbf{H} = \sigma\left(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right), \tag{3.8}$$

$$\mathbf{Y} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}. \tag{3.9}$$

More complex models can be constructed by continuously stacking the hidden layers on top of the other, such as $\mathbf{H}^{(1)} = \sigma_1\left(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)$ and $\mathbf{H}^{(2)} = \sigma_2\left(\mathbf{H}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}\right)$, and so on [41].

3.3. Convolutional Neural Networks

CNNs are a class of ANNs that are often used for processing data in a grid-like pattern such as 1D time-series data or 2D images. As the name suggests, they use "convolution" operation rather than conventional matrix multiplication in at least one of the model layers. The convolution operation allows the network to learn spatial hierarchies in the input data by applying a series of filters, which slide over the input to produce feature maps. These feature maps are then passed through several layers to capture complex patterns [40].

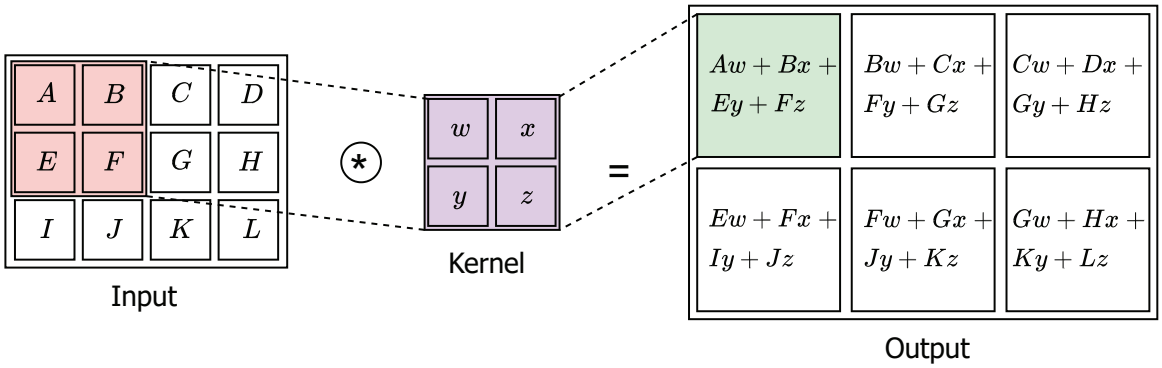

Figure 3.3. An example of 2D convolution as performed in convolutional neural networks.

In mathematics, convolution is an operation that combines two functions defined on real values to obtain a third function. Basically, convolution involves flipping one of the functions and shifting it across the other while multiplying the overlapping regions of these functions at each section. Consider two continuous functions $x(t)$ and $w(t)$, their convolution is obtained as

$$\begin{aligned} s(t) &= x(t) * w(t) \\ &= \int_{-\infty}^{\infty} x(a)w(t-a)\,da \end{aligned} \tag{3.10}$$

where $*$ denotes the convolution operation, t is a real-valued independent variable such as time and the integration is performed with respect to a dummy variable a. Now consider the case where functions $x[t]$ and $w[t]$ are defined only on integer t, their discrete convolution is defined as

$$\begin{aligned} s[t] &= x[t] * w[t] \\ &= \sum_{a=-\infty}^{\infty} x[a]w[t-a]. \end{aligned} \tag{3.11}$$

In deep learning, the function x represents the input and w denotes the kernel, which is also referred to as a filter. The input typically consists of a large multi-dimensional array of data whereas the kernel comprises a multidimensional array of parameters learned by the model. These arrays are referred to as tensors. Given a

2D image I as an input with a corresponding 2D kernel K, their convolution which involves kernel flipping is computed as

$$\begin{aligned} S(i,j) &= (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \\ &= (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \end{aligned} \tag{3.12}$$

However, for implementation, many deep learning libraries use cross-correlation in place of convolution as shown in Figure 3.3. This operation does not involve kernel flipping and is performed as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{3.13}$$

3.4. Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of ANNs that are specialized for processing a sequence of values such as time series data and converting them into a sequential data output [42]. Basically, every function that includes recurrence may be thought of as an RNN. Like feed-forward neural networks, RNNs use training data to learn in which the data moves via the hidden layer to obtain the output. Traditional feed-forward neural networks only employ data from the input node due to their lack of memory. In contrast, RNNs are capable of operating in both the input and internal states. Nevertheless, in real implementations, RNN models experience the problems of vanishing gradient and gradient explosion. To alleviate these problems, activation functions as described earlier may be used. An alternative way is to initialize the biases to zero and the weights to an identity matrix. The most reliable approach, however, is using nonlinear regulators, also known as "gates" such as the long short-term memory (LSTM) which also improves the model learning capability.

3.4.1. Long Short-Term Memory

The fundamental idea behind LSTM is its ability to retain pertinent information and forget the irrelevant ones through its memory cell. The computational unit helps to regulate information flow within the cell by selectively updating its internal state to produce an output illustrated in Figure 3.4. This controlled flow of data enables the model to capture long-term temporal dependencies, thereby significantly enhancing model performance on sequential tasks.

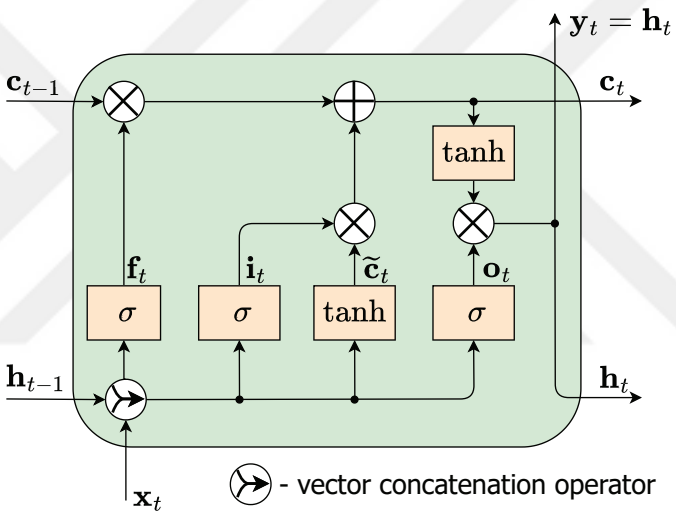

Figure 3.4. Internal structure of the long short-term memory cell.

LSTM consists of two main components as shown in Figure 3.4: the cell state $\mathbf{c}_t$ which stores information over time and the hidden state $\mathbf{h}_t$, which stores past information. At each time step t, the model selectively adds or removes information from these states. The input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, and output gate $\mathbf{o}_t$ are layers of learned nonlinear transformations that govern these updates. These gates receive inputs from the hidden state of the previous time step and the current sequence $\mathbf{x}_t$. The parameters $\mathbf{W}$ and $\mathbf{V}$ denote learned weight matrices, $\mathbf{b}$ represents the learned bias, σ denotes a logistic Sigmoid function in this case, and $\otimes$ signifies the Hadamard product (elementwise matrix multiplication). During training, these components learn which features are essential to retain and which to forget.

The cell utilizes both the current states and the input sequence to compute updated states. The cell input is recurrently connected to the hidden and cell states. Ultimately, the current hidden state serves as the cell output $\mathbf{y}_t$, which is then passed to the next layer to determine its gating mechanism. This process continues similarly until the last layer is reached [43] with updates performed as shown in the following equations

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right), \tag{3.14}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{V}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right), \tag{3.15}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{V}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right), \tag{3.16}$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{V}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right), \tag{3.17}$$

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{c}}_t, \tag{3.18}$$

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh\left(\mathbf{c}_t\right), \tag{3.19}$$

$$\mathbf{y}_t = \mathbf{h}_t. \tag{3.20}$$

3.5. Autoencoders

Autoencoders are a type of ANNs that are trained to attempt to reconstruct the input. They are designed such that they learn the representation of data by reducing it into a low-dimension latent space and then decoding it back to the original space. This process aims to reconstruct the input as accurately as possible while also learning important features or patterns within the data. The entire network is trained via backpropagation with the objective of minimizing the reconstruction error which is the difference between the input $\mathbf{x}$ and its reconstructed version $\hat{\mathbf{x}}$.

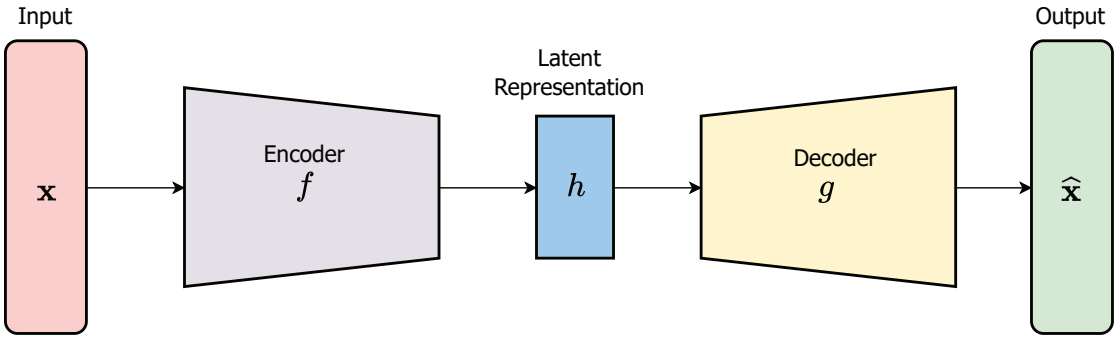

Figure 3.5. General structure of an autoencoder.

An autoencoder consists of an encoder which transforms the data into a lower dimension and is represented as

$$\boldsymbol{h} = f(\mathbf{x}) \tag{3.21}$$

where $\boldsymbol{h}$ is the hidden layer which represents the input in the latent space. Finally, the decoder which reconstructs the data is given by

$$\hat{\mathbf{x}} = g(\boldsymbol{h}). \tag{3.22}$$

Figure 3.5 illustrates the general configuration of an autoencoder, which transforms an input vector $\mathbf{x}$ into a reconstructed output $\hat{\mathbf{x}}$ via the hidden layer $\boldsymbol{h}$. It is important to note that autoencoders are designed not to perfectly copy the input i.e., $g(f(\mathbf{x})) = \mathbf{x}$, as doing so will not be useful. They are typically constrained to copy only an input that resembles the training data. This constraint helps the model to learn only the most important features of the input data [40].

3.6. Multi-Scale Convolutional Recurrent Encoder-Decoder Framework

Zhang *et al.* [44] proposed a Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) which is shown in Figure 3.6 to detect anomalies and provide a diagnosis of the root cause of the anomaly. MSCRED is a reconstruction-based method that uses the notion that previously unseen data or temporal patterns are poorly reconstructed and hence, tagged as anomalies.

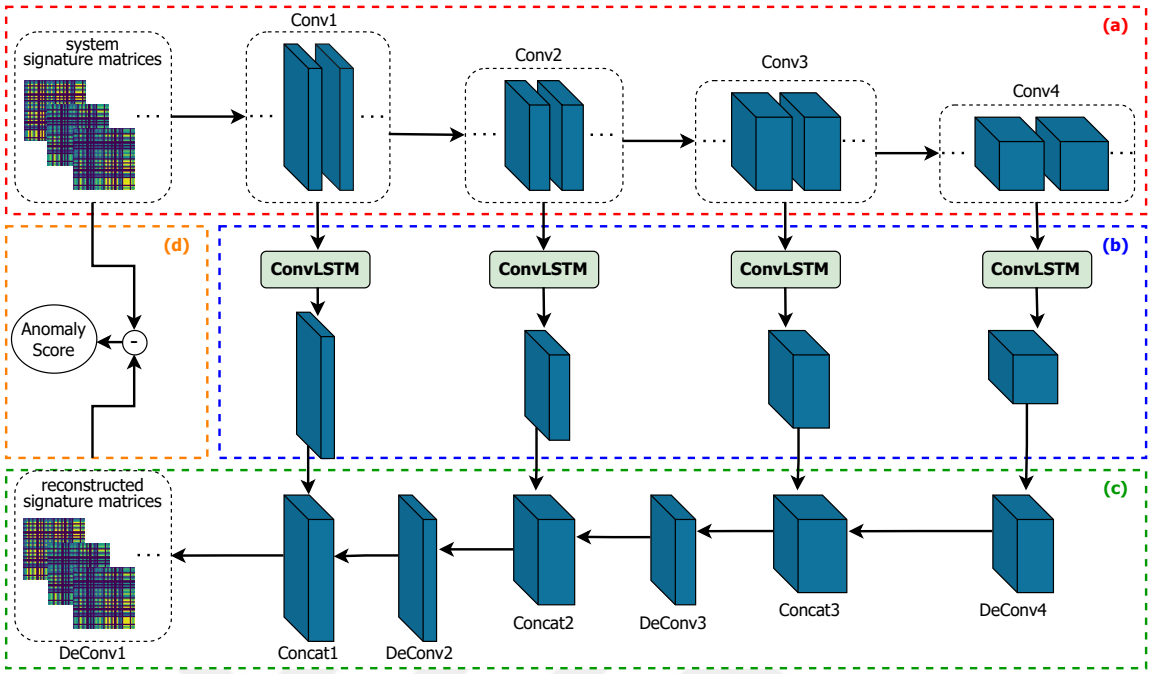

Figure 3.6. MSCRED framework: (a) Signature matrices encoding (b) Temporal pattern modeling (c) Signature matrices decoding (d) Anomaly detection.

First, MSCRED uses the idea of correlation within the MVTS data to construct system signature matrices (SSM) of the normal data to be used for training. SSM are a mathematical representation of the dataset which contain information about the correlations and temporal patterns present in the data [45]. These SSM are fed into a convolutional encoder for spatial pattern encoding [46]. Subsequently, the temporal patterns in the data are captured using an attention-based Convolutional Long-Short Term Memory (ConvLSTM) [47]. The attention mechanism enables the ConvLSTM to adaptively select the most relevant hidden states which enhances the learning dynamics of the model. It does so by selecting the most relevant prior states for temporal pattern modeling. Thereafter, a convolutional decoder is used to reconstruct the SSM. The convolutional encoder and decoder are implemented using the SELU activation function.

Hallac *et al.* [45] introduced the concept of Toeplitz Inverse Covariance-Based Clustering for MVTS data which uses the correlations to represent each cluster. This research highlighted the importance of correlations in MVTS which justifies the use

of SSM in representing the data. Consider two time series between the time windows $t-\omega$ and t represented as

$$\mathbf{x}_i^{\omega} = [x_{i,t-\omega+1}, x_{i,t-\omega+2}, \cdots, x_{i,t}]^{\mathrm{T}} \in \mathbb{R}^{\omega}, \tag{3.23}$$

$$\mathbf{x}_j^{\omega} = [x_{j,t-\omega+1}, x_{j,t-\omega+2}, \cdots, x_{j,t}]^{\mathrm{T}} \in \mathbb{R}^{\omega} \tag{3.24}$$

where $\mathbf{x}_i^{\omega}$ and $\mathbf{x}_j^{\omega}$ represent the data from the i-th and j-th sensors, respectively. The inner product of these time series is calculated as

$$m_{ij}^t = \frac{\left\langle \mathbf{x}_i^{\omega}, \mathbf{x}_j^{\omega} \right\rangle}{f} \tag{3.25}$$

where f is a scaling factor, $f = \omega$ in this design. The correlation matrices are computed as

$$\mathbf{M}^t = \begin{bmatrix} m_{11}^t & m_{12}^t & \cdots & m_{1n}^t \\ m_{21}^t & m_{22}^t & \cdots & m_{2n}^t \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1}^t & m_{n2}^t & \cdots & m_{nn}^t \end{bmatrix} \tag{3.26}$$

where the entries m_{ij}^t are the obtained inner products. To capture multi-scale inter-sensor correlations, three window sizes are used to represent short, medium, and long-term correlations. The correlation matrices $\mathbf{M}^t$ are concatenated with these windows to obtain SSM.

In reconstruction, the last hidden state of the ConvLSTM layer is fed in reverse order to the decoder, generating output with the same dimension as the input SSM. A square loss function is employed to minimize the objective function, which are the reconstruction errors calculated as

$$\mathcal{L} = \sum_{c=1}^{s} \left\| \mathcal{X}_{:,:,c}^t - \hat{\mathcal{X}}_{:,:,c}^t \right\|_2^2, \quad t = 1, 2, \ldots, N \tag{3.27}$$

where $\mathcal{X}_{:,:,c}^t$ and $\hat{\mathcal{X}}_{:,:,c}^t$ represent channel c of the input and reconstructed SSM, respectively and N is the time series length. The residual SSM which are used for anomaly detection are obtained by subtracting the reconstructed signature matrices from the

original input SSM. The number of poorly reconstructed entries of the signature matrices are used as the anomaly score. A reconstruction error threshold θ_R is defined as the cutoff value to determine the number of poorly reconstructed SSM. An anomaly threshold θ_A which optimizes the precision and recall to maximize the F1 score is determined experimentally. This framework is used in this thesis and enhanced using CL as explained later.

MSCRED is trained using Adaptive Moment estimation (Adam) [48] optimization algorithm. The experimental results show that MSCRED performs good in anomaly detection in comparison to a number of state-of-the-art methods. This framework is further evaluated using various datasets in a comprehensive MVTS anomaly detection study by Wagner *et al.* [33]. The results show that even though it performs good in several datasets, it is not universal as it performed relatively poor in some datasets.

3.7. Curriculum Learning

The fundamental idea behind CL is introducing ML models to progressively harder examples throughout the training process with the aim of making them learn more rapidly, improve their efficiency, and accuracy.

The concept of providing the input to the ML algorithm in a specified order does provide some benefits but not always as previous studies have shown [18, 49]. The CL training process starts by using smaller and easier dataset and gradually increasing the size and difficulty until the whole training dataset is utilized. This technique can enhance the convergence rate and performance in comparison to directly employing the whole training dataset.

According to Bengio *et al.* [18], the foundation of CL is based on identifying training criteria that constitute a curriculum. A curriculum, $C = \langle Q_1, \ldots, Q_k, \ldots, Q_K \rangle$ with training data $\mathbf{X}_{train}$ divided into K subsets is a sequence of training criteria Q_k which is a reweighted version of the targeted distribution $P(\mathbf{x})$ and is expressed as

$$Q_k(\mathbf{x}) \propto W_k(\mathbf{x})P(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbf{X}_{train} \tag{3.28}$$

where $W_k(\mathbf{x})$ is the weight and the following conditions have to be satisfied:

(i) $H(Q_k) < H(Q_{k+1})$, where $H(Q_k)$ is the entropy,
(ii) $W_k(\mathbf{x}) \leq W_{k+1}(\mathbf{x})$,
(iii) $Q_K(\mathbf{x}) = P(\mathbf{x})$.

The first condition implies that the variety and information of the chosen training criterion should progressively increase, as such the entropy should increase as well. The likelihood of sampling harder training data increases as the training progresses. In the second condition, more data is progressively added, hence, the weight should also increase. The third condition indicates that in the final training round, all the training data is used.

3.7.1. Curriculum Design

In order to design a CL framework, there are three essential questions that need to be addressed:

(i) What criteria are used to arrange the training data in order of difficulty?
(ii) What criteria are used to arrange the deep learning algorithm in order of model complexity?
(iii) When should the deep learning algorithm be presented with more complex data during training?

The above questions can be answered by defining two fundamental components. For the first and second questions, a "difficulty measurer" to sort the data starting from easy to hard or the models from simple to complex has to be defined. For the third question, a "training scheduler", also known as pacing function, has to be defined to sample the training data to be presented to the algorithm at each round. As the training progresses, the training scheduler decides on the data to be fed to the algorithm

until the whole data is sampled. A training scheduler can, for example, start with a quarter of the easy data in the dataset and have uniform increments at each round until all the training data is used. Accordingly, a suitable curriculum that can produce significant benefits can be designed.

After addressing the above questions, there have to be ways of designing the aforementioned components that facilitate CL. One way is for a human to manually design both the difficulty measurer and training scheduler, a criterion termed predefined CL. An alternative approach would be to employ data-driven algorithms in designing these two components, an approach denoted as automatic CL.

3.7.2. Predefined Curriculum Learning

3.7.2.1. Predefined Difficulty Measurers. A number of predefined difficulty measurers have been designed by researchers across different fields that employ ML. Several of the designed difficulty measurers are in the computer vision and natural language processing fields. Most of the approaches are based on the structural complexity, distributional diversity, and noise estimation of the data.

Structural complexity considers the dimension of the data, for example, the text length can be considered to be a difficulty measurer with longer texts being considered as the harder data. Other components that constitute a sentence such as phrases and grammar among others can also be used in natural language processing tasks [50]. In computer vision tasks, the number of objects in an image can be taken into consideration. In face recognition, head pose angles can be used as the difficulty measure in which the images facing the camera are considered easy and the side profile images are in the hard group [19]. Distributional diversity means the variety of elements of the data. For instance, the diversity of the vocabulary, where the rare words are considered to constitute the more complex data group. Noise estimation approximates the level of noise that the training data contains. For example, signal to noise ratio can be used to determine the noise level in audio data, where clean data makes up the easy group.

3.7.2.2. Predefined Training Schedulers. Unlike predefined difficulty measurers which highly vary with respect to the data, predefined training schedulers can be classified into continuous and discrete. Continuous training schedulers select the consecutive dataset difficulty to be used in training after each round whereas discrete schedulers adjust the dataset after a fixed number of rounds greater than one or after a specified convergence criteria.

Owing to the efficiency and simplicity of discrete schedulers, they are the commonly used type. The Baby Step technique [51], a widely employed discrete scheduler in CL initially splits the sorted data starting from less complex to more complex data. As training progresses, the subsequent split is integrated into the training examples after a predetermined number of rounds. Once all of the splits have been combined and employed in the model training, depending on the chosen stopping criterion, the training either ends or proceeds for a number of additional epochs. Further examples of discrete training schedulers include One-Pass and variants of the Baby Step [18].

Predefined CL techniques are easy to employ and effective, however, there are some drawbacks to using them. It is an intensive and challenging task to manually choose the difficulty measurer and training scheduler that are best suited for a particular application. Furthermore, expert knowledge is required in designing these components. Even so, in some tasks with high dimensional features, it is very difficult even for the expert to manually define these components. A further limitation is that it is difficult to find the optimal hyperparameters of the training scheduler such as the learning rate which may affect the performance of the model.

3.7.3. Automatic Curriculum Learning

In the scope of predefined CL, a human expert serves as the teacher to the model which is the student or learner. Automatic CL seeks to minimize the reliance on a human expert as the teacher. The three automatic curriculum techniques are Self-Paced Learning (SPL), transfer teacher, and reinforcement learning teacher.

SPL allows the student model to take on the role of a teacher and evaluate the complexity of the training data based on its experiences. Since the student model chooses its own learning pace based on its experience, this method is comparable to how actual students gain knowledge on their own. In SPL, the order in which the data samples are presented to the model is not known but is determined in relation to the performance of the model. In fact, the complexity is constantly assessed throughout training, hence, the order in which the samples are fed to the model may constantly change. In ML environment, the notion of SPL was originally proposed by Kumar *et al.* [52], in which the ML model is trained at each epoch using the subset of the training set that achieves the least loss [21, 53].

In SPL, the student model acts as the teacher model and determines the difficulty measurer automatically on its own. Nevertheless, this approach is subject to limitations at the start of the training when the model has not learned yet. At this stage, the model would have gained too little experience to be capable of deciding an efficient difficulty measurer. Consequently, an intuitive approach would be to utilize an experienced teacher to assist the student in evaluating the data and creating a curriculum. This is a technique known as transfer teacher or teacher-student [53]. Transfer teacher frameworks employ a pre-trained teacher model to determine the complexity of the training data subject to how well the teacher model performs and transfers this knowledge to the student. Then the data is arranged to constitute a curriculum and fed to the student model according to the difficulty.

Reinforcement learning teacher is an interactive framework that allows the teacher to adaptively select data based on the input of the student. Reinforcement learning is a branch of ML that focuses on training computers how to make decisions in uncertain environments. It consists of an agent that learns how to make decisions subject to the rewards or penalties it receives. The objective of reinforcement learning is to find the optimal strategy that maximizes the cumulative reward of the agent [54]. In a reinforcement learning environment, the performance of the student model improves owing to the customized training samples chosen by the teacher model. At the same time,

the teacher model also successfully modifies the curriculum such that the performance of the student continues to improve.

In this thesis, predefined CL is used because part of the study is to analyze the most important data feature that can lead to improved anomaly detection performance. SPL considers each data individually and determines its difficulty, before sorting it. This may lead to the destruction of the patterns as consecutive data points may have different difficulty levels. Moreover, SPL is generally based on the loss values, therefore, neglecting prior knowledge from an expert may result in poor sorting of the data as ordering based on some features has to be performed manually via statistical tests which SPL is incapable of performing. Inspired by the transfer teacher technique, this thesis also seeks to investigate the model-based CL which transfers the learned model parameters instead of the data difficulty to the subsequent model.

3.8. Evaluation Metrics

Precision, recall, and F1 score are used to evaluate the model performance. Precision measures the proportion of correctly identified anomalies whereas recall measures the ratio of actual anomalies that the model successfully detected. The F1 score is a harmonic mean of precision and recall. These metrics are calculated as

$$Precision = \frac{TP}{TP + FP}, \tag{3.29}$$

$$Recall = \frac{TP}{TP + FN}, \tag{3.30}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.31}$$

where TP, FP, and FN are the number of true positive, false positive, and false negative classifications, respectively, as shown in Figure 3.7. They are used to assess the extent to which the model detects anomalies in a dataset and are defined as follows:

- TP represents instances in which anomalies are correctly detected by the model. In other words, the model predicts that an instance is an anomaly, and it is indeed

an anomaly according to the ground truth.

- FP denotes normal data which the model incorrectly classifies as anomalies. The ground truth indicates that an instance is normal while the model predicts it to be an anomaly. FP is also known as a Type I error.
- FN are cases in which anomalies are incorrectly classified as normal instances by the model. The model predicts that an instance is normal, but it is actually an anomaly according to the ground truth. FN is also known as a Type II error.

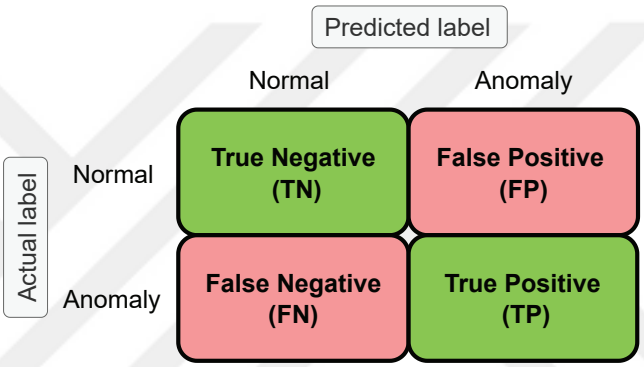

Figure 3.7. Confusion matrix for anomaly detection evaluation.

3.9. Summary of the Chapter

In this chapter, deep learning methods relevant to this thesis were presented. First, an overview of ANNs, namely CNNs, LSTM, and autoencoders was provided. This chapter also highlighted the importance of activation functions such as introducing nonlinearity. Then a detailed explanation of how the aforementioned ANNs were used to design MSCRED which was later improved using CL in Chapter 4 was presented. Moreover, the method of representing MVTS data as signature matrices which has proven effective in learning temporal patterns was explained. Different approaches to designing a curriculum and the reasons for the choice of the selected CL technique were given. Finally, the evaluation metrics used to analyze the efficacy of designed models were provided. The remaining chapters build upon the techniques explained in this chapter by proposing novel CL-based training strategies for MVTS data.

4. ANOMALY DETECTION IN WATER TREATMENT PROCESSES

Water treatment plants have become more complex, hence several monitoring and diagnostic systems are required to analyze operational data for anomaly detection. This chapter presents the proposed CL-based anomaly detection frameworks and evaluates them on a benchmark water treatment dataset.

4.1. Secure Water Treatment Dataset

Secure Water Treatment (SWaT) which consists of cyber and physical attacks is a popular dataset in MVTS anomaly detection [55, 56]. A cyber attack is launched via a communication network whereas a physical attack is an anomaly on a physical device such as a motor which may lead to disruption of the process. The SWaT testbed which consists of six stages is a scaled-down version of an actual industrial water treatment plant. The SWaT testbed setup follows a scheme shown in Figure 4.1 where the higher level controller consists of human-machine interfaces, SCADA workstation, and historian among other systems, and the low level controller consists of programmable logic controllers (PLCs). It also consists of two communication networks: Level 0 which allows each PLC to communicate with its sensors and actuators and Level 1 which facilitates communication between the PLCs and the higher level controller.

The purpose of this dataset is to facilitate research in ICS security by understanding the effects of anomalies, evaluating the extent of cyber and physical attacks on ICS, assessing the performance of anomaly detection frameworks, and determining the efficiency of defense mechanisms. The data was collected in December 2015 over a period of eleven days with the first seven days under normal operation and attacks injected in the remaining four days. The duration of the attacks was varied from a few minutes to an hour. In this thesis, the dataset which consists of only physical attacks is used and part of it is shown in Figure 4.2.

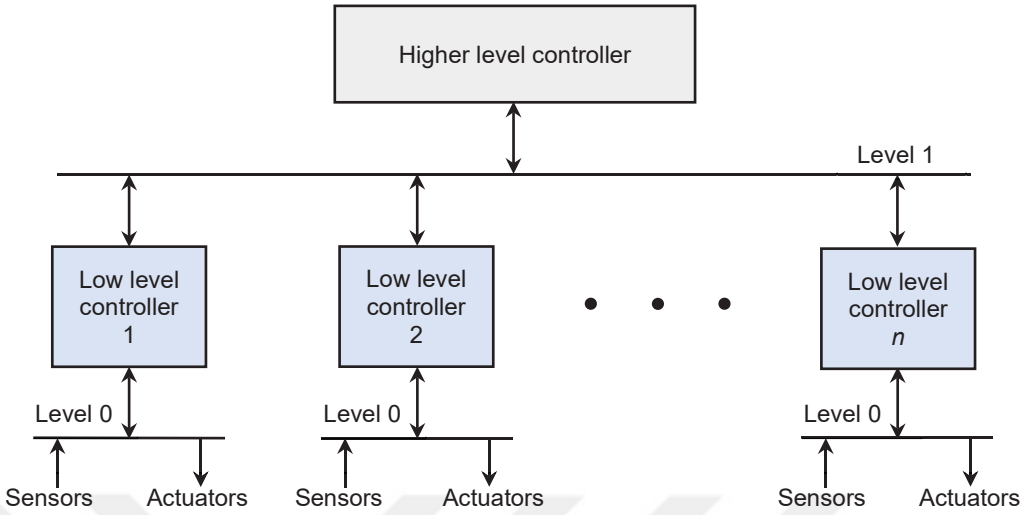

Figure 4.1. Hierarchical control scheme.

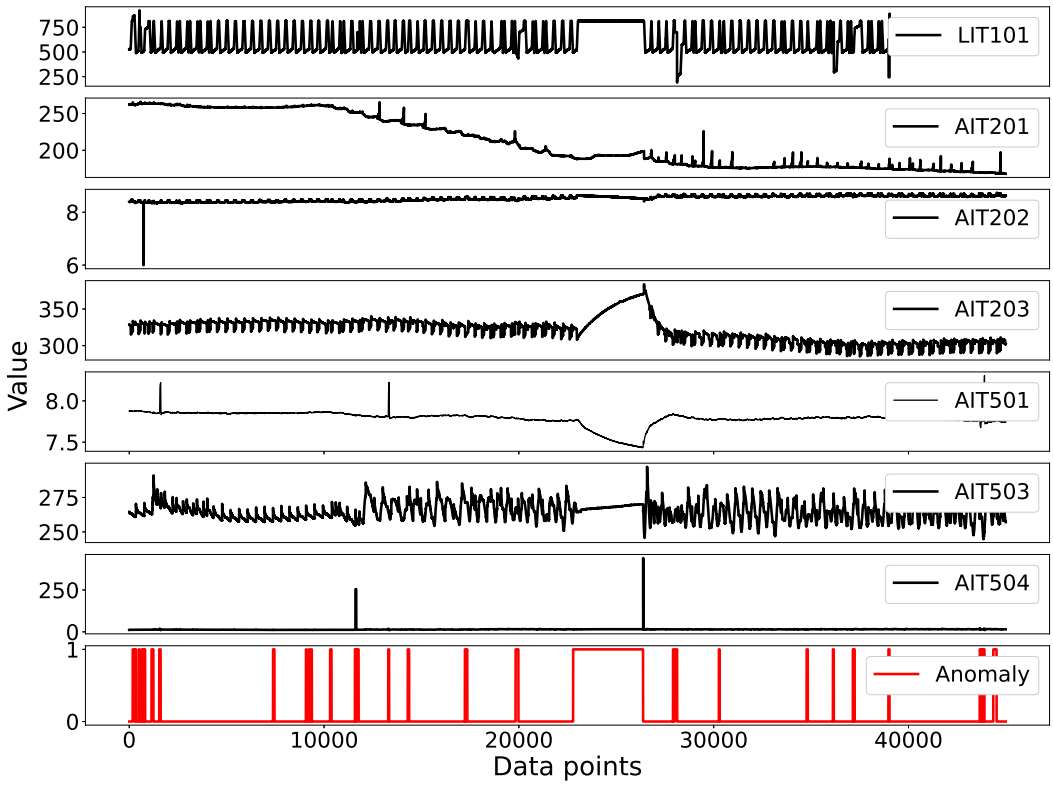

Figure 4.2. Examples of time series data from the downsampled SWaT test set and labeled anomaly locations.

4.2. Proposed Data-based Curriculum Learning for Anomaly Detection

Time series data contains various features such as temporal complexity, sequence length, seasonality, and trends that change over time. These features can influence the anomaly detection performance of a deep learning model. The data can be organized starting with simpler regions first, then gradually introducing more complicated and unpredictable sections. Effective pattern learning can also be achieved by training the algorithm starting with shorter sequences and progressively lengthening them until the whole dataset is utilized. Furthermore, seasonality and long-term trends in the data, which can be more challenging for time series models to capture can be introduced in the later training stages of the model. Since data is used to design a curriculum, this approach is called Data-based CL (DCL) for anomaly detection [1] and is shown in Figures 4.3 and 4.4.

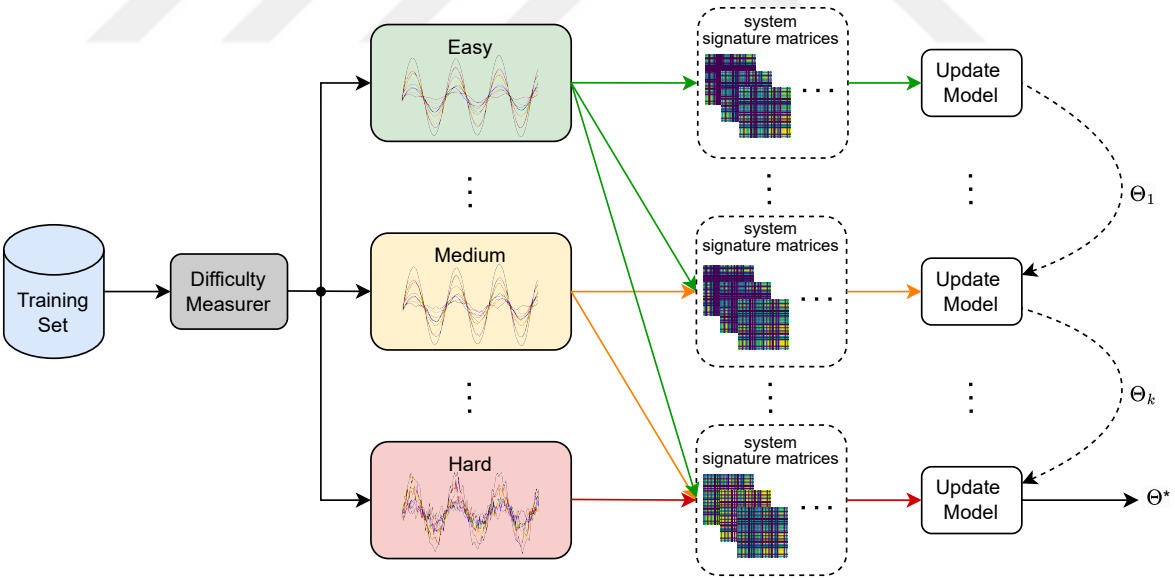

Figure 4.3. Proposed data-based curriculum learning for multivariate time series anomaly detection framework.

Inputs: $\mathbf{X}_{train}$: Training set; $\mathbf{X}_{test}$: Test set; $\mathbf{X}_{CL}$: Curriculum training set; C: Difficulty measurer; θ_A: Anomaly threshold.
Outputs: Θ^*: Optimal model; A: Anomaly score.
$\mathbf{X}'_{train} = \text{sort}(\mathbf{X}_{train}, C)$;
$\left\{\mathbf{X}^1_{train}, \mathbf{X}^2_{train}, \cdots, \mathbf{X}^K_{train}\right\} = \mathbf{X}'_{train}$ where $C(\mathbf{x}_a) < C(\mathbf{x}_b)$, $\mathbf{x}_a \in \mathbf{X}^i_{train}$, $\mathbf{x}_b \in \mathbf{X}^j_{train}, \forall i < j$;
$\mathbf{X}_{CL} = \emptyset$;
for $k = 1 \cdots K$ **do**
    $\mathbf{X}_{CL} = \mathbf{X}_{CL} \cup \mathbf{X}^k_{train}$;
    Generate system signature matrices from $\mathbf{X}_{CL}$;
    while not converged for E epochs **do**
        train$(\Theta, \mathbf{X}_{CL})$;
    end while
end for
Generate system signature matrices from $\mathbf{X}_{test}$;
for test data points $\mathbf{x}_{test} \in \mathbf{X}_{test}$ **do**
    test$(\mathbf{x}_{test}, \Theta^*)$;
    if $A(\mathbf{x}_{test}) > \theta_A$ **then**
        $\mathbf{x}_{test}$ has an anomaly;
    end if
end for

Figure 4.4. Data-based curriculum learning for multivariate time series anomaly detection.

4.2.1. Curriculum Design Based on Sequence Length

Since the time series length can be varied, it may be used as a difficulty measure. This can be done by feeding the model with progressively increasing sequence length which can enhance effective pattern learning by the model [57]. This design is inspired by the observation that traditional ML models often face difficulties in gen-

eralizing effectively when trained on extensive datasets. The improvement of deep learning models has been driven by three main factors, namely increased model capacity, improved computational power, and the availability of large datasets. Sun *et al.* [58] explored the importance of data size in deep learning by conducting computer vision experiments and they concluded that larger datasets improve representation learning. This suggests that utilizing large datasets can potentially increase the performance of deep learning models. Further research by Brigato and Iocchi [59] showed that training high-capacity models with small datasets results in significantly lower performance compared to utilizing large datasets. This proves that dataset size affects model learning and performance and, hence, can be utilized to design a curriculum. The SWaT dataset is segmented percentage-wise and the MSCRED model is trained with progressively increasing sequence length as follows:

- DCL-MSCRED-P2: DCL for MSCRED based on a dual percentage-wise split. Initially, the MSCRED model is trained on 50% of the dataset. Next, in the final training round, it is trained on the entire dataset.
- DCL-MSCRED-P3: DCL for MSCRED based on a ternary percentage-wise split. The MSCRED model is initially trained on approximately 33% of the data. Subsequently, it is trained on about 67% of the dataset and then on the entire dataset in the final training round.

4.2.2. Curriculum Design Based on Window Size

In this method, the window size of the data during training is used as the difficulty measure. For reconstruction-based frameworks, the window size is defined as the number of data points used as input features to reconstruct the subsequent data point. Since it is easier to reconstruct data that is close to the initial data point within the window, a shorter window is considered to be easy to learn. However, as the window size increases, data becomes more variable, and trends and temporal patterns may change considerably, making reconstruction more challenging. In this approach, the whole training set is used at each round with different window sizes as follows:

- DCL-MSCRED-W2: DCL for MSCRED using two different window sizes. Initially, the MSCRED model is trained with a short window, $\omega = 10$ data points (100 seconds). Then in the final training round, it is trained with short and long windows, $\omega = \{10, 60\}$.
- DCL-MSCRED-W3: DCL for MSCRED using three different window sizes. The MSCRED model is first trained with a short window, $\omega = 10$. Subsequently, a medium window is introduced, $\omega = \{10, 30\}$. Finally, the model is trained with short, medium, and long windows, $\omega = \{10, 30, 60\}$.

4.2.3. Curriculum Design Based on Time Series Stationarity

In this approach, it is proposed to use the time series stationarity as the difficulty measure. Stationarity means that the statistical properties of the time series such as the mean, variance, covariance, and standard deviation do not change with time. The inverse is true for a non-stationary process. Due to the constant nature of the statistical moments of a stationary process, it is easier to forecast or reconstruct the data. On the other hand, the dynamic nature of the statistical moments and variations in data distributions in non-stationary time series data makes it more difficult to predict or reconstruct. Therefore, the stationarity of the time series can be used as a difficulty measure. For MVTS data, model training starts with more stationary time series and progressively includes less stationary data from multiple sensors. The Augmented Dickey-Fuller (ADF) test is employed to measure the stationarity of each time series [60].

4.2.3.1. Augmented Dickey-Fuller Test. The ADF test is used to determine the presence of a unit root in time series data. A unit root means that the time series has a root of one when modeled as a stochastic process. If a time series has a unit root, it is considered non-stationary, which indicates that its statistical characteristics, such as mean and variance, change with time. The null hypothesis, H_0 and alternative hypothesis, H_1 are represented as follows:

H_0: Time series has unit root test, hence non-stationary,

H_1: Time series has no unit root test, hence stationary.

Given a sequence of time series data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, the ADF test can be employed to determine the stationarity of each process. This test uses the ordinary least squares estimator $\widehat{\rho}_n$ of ρ to determine the existence of a unit root. This estimator is attained from fitting the regression equation to the time series data. The regression equation is expressed as

$$\mathbf{x}_t = \rho \mathbf{x}_{t-1} + \sum_{j=1}^{p} a_{j,p} \Delta \mathbf{x}_{t-j} + e_{t,p} \tag{4.1}$$

where $\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$, p is the order which depends on n, i.e., p_n. In the null hypothesis $H_0 : \rho = 1$, the common assumption is that $\mathbf{x}_t$ is obtained by integrating a linear, infinite order autoregressive process and can be represented as

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{u}_t, \quad t = 1, 2, \ldots, \tag{4.2}$$

where $\mathbf{x}_0 = 0$ and

$$\mathbf{u}_t = \sum_{j=1}^{\infty} a_j \mathbf{u}_{t-j} + e_t \tag{4.3}$$

where $\{e_t\}$ is a sequence of independent and identically distributed random variables which have zero mean and variance $0 < \sigma_e^2 < \infty$. The stationarity and causality of the sequence $\{\mathbf{u}_t\}$ is ascertained by assuming that $\sum_{j=1}^{\infty} |j|^s |a_j| < \infty$ for some $s \geq 1$ and $\sum_{j=1}^{\infty} a_j z^j \neq 0$ for all $|z| \leq 1$.

To test the null hypothesis, Dickey and Fuller [60] proposed the following test statistic

$$t_n = \frac{\hat{\rho}_n - 1}{\widehat{std}\,(\widehat{\rho}_n)} \tag{4.4}$$

where $\widehat{std}\,(\widehat{\rho}_n)$ represents an estimator of the standard deviation of the ordinary least squares estimator $\widehat{\rho}_n$ [61]. The ADF test statistic is used to determine whether the data is stationary or not. If the test statistic is more negative than the critical value, the time

series is more stationary as shown in Figure 4.5. However, if the test statistic is less negative, the null hypothesis is rejected and it can be concluded that the time series has a unit root. This test effectively determines if the data has a trend but it fails to reject the null hypothesis of a unit root if the time series consists of a sequence of constant values. In this case, the ADF test statistic would likely be high, indicating that the series is less stationary as shown in Figure 4.6. This is due to the lack of variability or trend for analysis, which makes it challenging to differentiate between an underlying pattern and a random fluctuation. The time series is arranged in order of increasing ADF test statistic value and the following curriculum designs are proposed:

- DCL-MSCRED-S2: DCL for MSCRED based on two splits of the dataset using time series stationarity. The MSCRED model is initially trained with the first 25 time series (about 50%) of the ordered SWaT dataset which are more stationary and considered easy, and then the remaining hard 26 time series are included in the final training round.
- DCL-MSCRED-S3: DCL for MSCRED based on three splits of the dataset using time series stationarity. The MSCRED model is initially trained with the first 33% time series of the ordered SWaT dataset, then with 67%, and finally with the whole dataset.

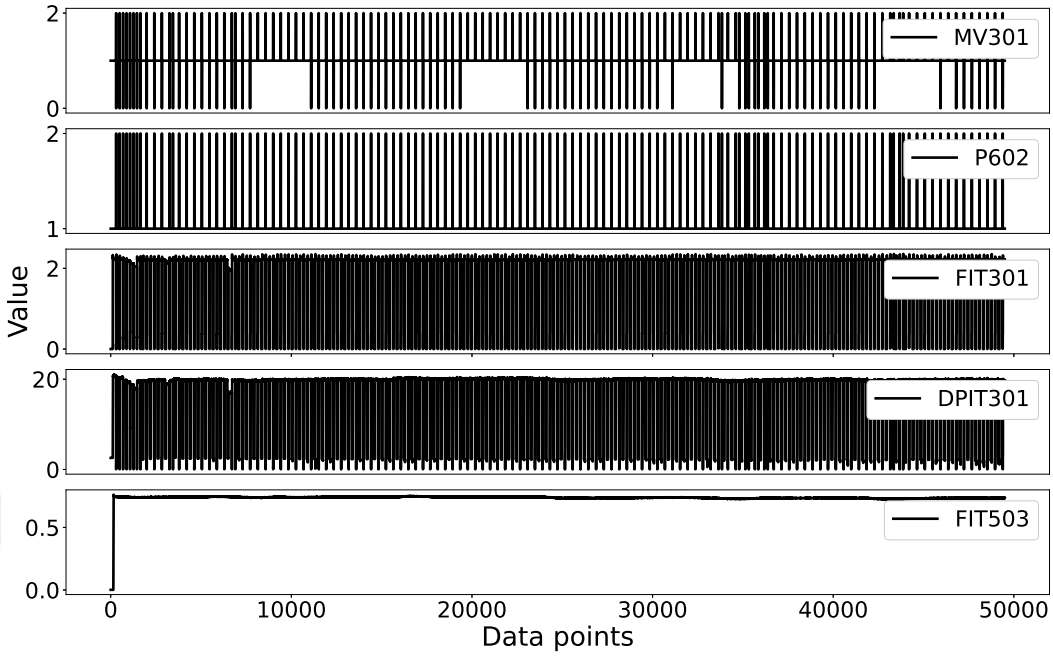

Figure 4.5. Some of the more stationary time series data of the downsampled SWaT training set.

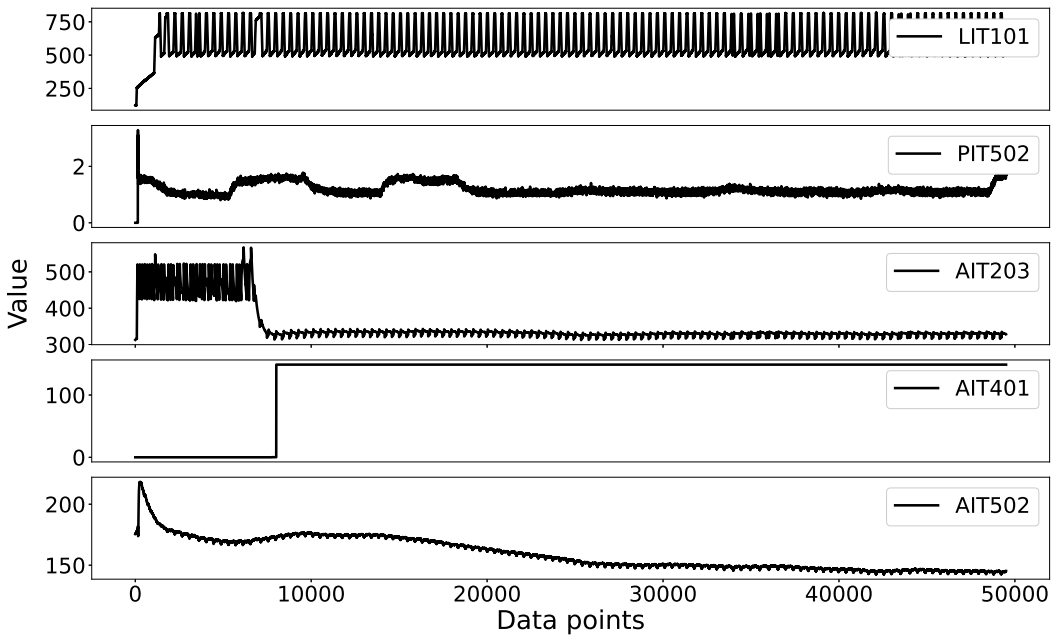

Figure 4.6. Some of the less stationary time series data of the downsampled SWaT training set.

4.2.4. Curriculum Design Based on Data Noise Level

This curriculum design is inspired by continuation methods which are optimization techniques used in non-convex functions [18]. A sequence of objective functions is created taking into account the difficulty to find the minima. A smooth function, which is the easiest to find the minima is considered first, and then progressively more complex functions are optimized. At each stage, the solution of the simplified problem is used as the initial guess for the more complex problem. Ultimately, the original objective function is optimized. Since CL follows a similar learning criterion, it can be thought of as a continuation method. Continuation methods direct the search for the minima toward the parameter space with a high likelihood of containing the minima. By applying a continuation method, it is possible to solve non-convex optimization problems that would be difficult or impossible to solve with traditional optimization techniques. This method helps solve problems with several local minima or complex geometries by allowing the optimization algorithm to escape from local minima and find the global minimum.

Smoothing is a technique used to reduce noise and variability in data. In MVTS data such as the columns of the SWaT dataset, smoothing is used to remove short-term fluctuations and highlight long-term trends. In this research, the Savitzky-Golay (S-G) filter [62] is employed to generate smoother versions of the data for the initial training rounds of the proposed framework. This is a type of linear filter that performs convolution on the data by using local polynomial regression which smooths the data using the least squares approximation. It is commonly used to smooth noisy data, particularly in tasks involving data analysis and signal processing. Consider the following signal

$$x[nT] = s[nT] + w[nT], \quad n = 1, 2, 3, \ldots, N \tag{4.5}$$

where T is the sampling period, N is the number of data points, s_n is the nth sample of a time-varying signal $s(t)$, and w_n denotes white Gaussian noise with mean zero and variance σ^2, and n refers to the time. For simplicity, the notation a_n represents the nth sample of signal a, i.e., $a_n \triangleq a[nT]$. The aim is to fit a polynomial $q_n = \sum_{k=0}^{p} a_k (nT)^k$

in the least squares sense to the time series data and minimize the following objective function

$$\mathcal{C}_p = \sum_{n=-\frac{\omega}{2}}^{\frac{\omega}{2}} (q_n - x_n)^2 = \sum_{n=-\frac{\omega}{2}}^{\frac{\omega}{2}} \left(\sum_{k=0}^{p} a_k (nT)^k - x_n \right)^2 . \tag{4.6}$$

The data point to be smoothed is symmetrically positioned around a window from $-\frac{\omega}{2}$ to $\frac{\omega}{2}$. When a polynomial is fitted, the smoothed output value can be determined using the data points within the defined window. The smoothed output value at any other time can be calculated by centering the data point under consideration. The main parameters of this filter are:

- Window length (ω): This parameter specifies how many neighboring points are considered for smoothing. For more smoothing, a larger window length has to be used.
- Polynomial order (p): Higher polynomial orders are capable of capturing more noisy and complex patterns.

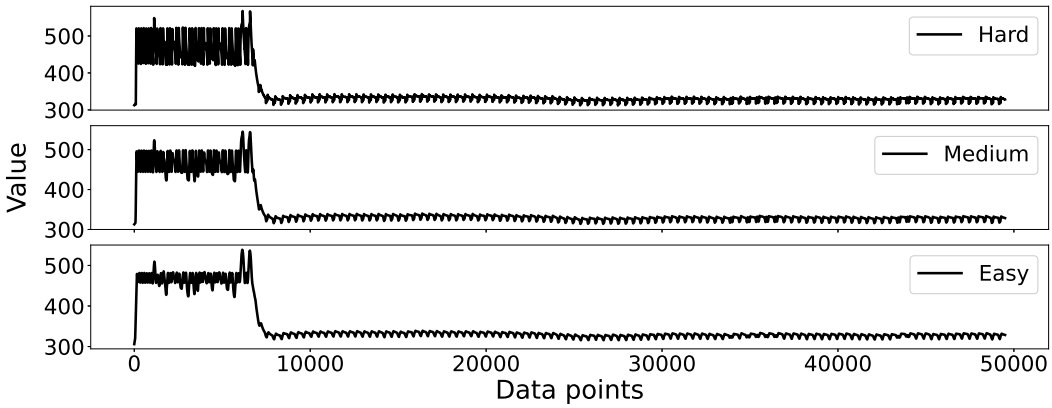

Figure 4.7. Easy, medium, and hard time series data of the AIT-203 sensor of the downsampled SWaT training set.

The purpose of this technique is to investigate the effect of noise level on the data by gradually introducing noise during training. Two different levels of smoothing are performed which are single smoothing in which the S-G filter is applied once to each time series data and double smoothing where the filtering is applied twice. Double

smoothing further reduces short-term fluctuations and emphasizes long-term trends compared to single smoothing as shown in Figure 4.7. The following approaches are investigated with $\omega = 50$ and $p = 1$:

- DCL-MSCRED-N2: DCL for MSCRED based on two different noise levels in the data. The MSCRED model is initially trained with the double-smoothed (easy) data and then with the original data (hard) in the final round.
- DCL-MSCRED-N3: DCL for MSCRED based on three different noise levels in the data. The MSCRED model is first trained with the double-smoothed (easy) data, followed by the single-smoothed data (medium), and finally with the original data (hard).

4.3. Proposed Model-based Curriculum Learning for Anomaly Detection

In research, CL has only been limited to data, however, since different anomaly detection models have different complexity, CL may be extended to the models. Training can be started with simpler modeling techniques and more complex algorithms or architectures can be utilized later. This thesis seeks to investigate a model-based approach by sorting different algorithms from "simple" to "complex" as shown in Figures 4.8 and 4.9. Since deep learning frameworks vary in architecture, model size, and optimization process, model complexity is used as the difficulty measure [1]. The proposed Model-based CL for Anomaly Detection (MCLAD) is designed as follows:

- MCLAD: It is a very difficult task to analytically determine the model complexity of deep learning techniques particularly if the models are made up of distinct ANNs [63]. Consequently, in this design, the LSTM-ED [64] which has an architecture designed with the main ANNs used in MSCRED is utilized. Since MSCRED incorporates CNNs and an attention mechanism along with the LSTM in its architecture, its design is inherently more complex compared to LSTM-ED. Also, these models are evaluated independently after training with the whole training set. As expected, the higher capacity MSCRED outperforms the simple

LSTM-ED which further justifies the selection of the MSCRED as the complex model. The LSTM-ED model is trained with the SSM from the whole training set and the model parameters are used to initialize MSCRED which is also trained on the whole training set.

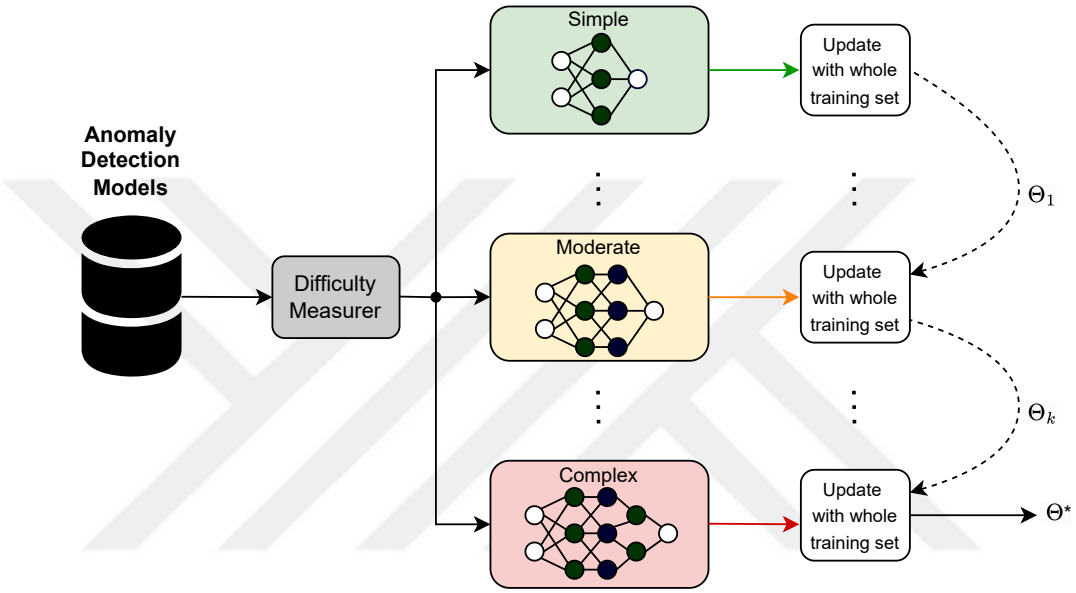

Figure 4.8. Proposed model-based curriculum learning for multivariate time series anomaly detection framework.

Inputs: $\mathbf{X}_{train}$: Training set; $\mathbf{X}_{test}$: Test set; C: Difficulty measurer; $\{\Theta^a, \Theta^b, \cdots, \Theta^z\}$: Anomaly detection models ordered in increasing complexity, $C(\Theta^a) < C(\Theta^b) <, \cdots, < C(\Theta^z)$; θ_A: Anomaly threshold.
Outputs: Θ^*: Optimal model; A: Anomaly score.
Generate system signature matrices from $\mathbf{X}_{train}$;
for each model $\Theta \in \{\Theta^a, \Theta^b, \cdots, \Theta^z\}$ **do**
  while not converged for E epochs **do**
    train($\Theta, \mathbf{X}_{train}$);
  end while
  Set Θ as initial model for subsequent training round;
end for
Generate system signature matrices from $\mathbf{X}_{test}$;
for test data points $\mathbf{x}_{test} \in \mathbf{X}_{test}$ **do**
  test($\mathbf{x}_{test}, \Theta^*$);
  if $A(\mathbf{x}_{test}) > \theta_A$ **then**
    $\mathbf{x}_{test}$ has an anomaly;
  end if
end for

Figure 4.9. Model-based curriculum learning for multivariate time series anomaly detection.

4.4. Proposed Data-based and Model-based Curriculum Learning for Anomaly Detection

Brigato and Iocchi [59] observed that in addition to the significance of large datasets for high-capacity models, simpler models actually perform better with smaller datasets compared to their complex counterparts. This insight motivated the design of a framework that combines data-based and model-based CL approaches. It is proposed to train a simple model with a smaller dataset, then use the obtained model parameters to initialize the complex model, which is then trained with the full dataset [1]. The

proposed Data and Model-based CL for Anomaly Detection (D-MCLAD) is designed as follows:

- D-MCLAD: Since the aim of this approach is to observe the effects of the data size, it is proposed to use the simplest way of obtaining a smaller dataset, which is a percentage-wise split. The LSTM-ED framework is initially trained on 50% of the training set. Subsequently, MSCRED is trained on the entire training dataset and evaluated on the test set as shown in Figure 4.10.

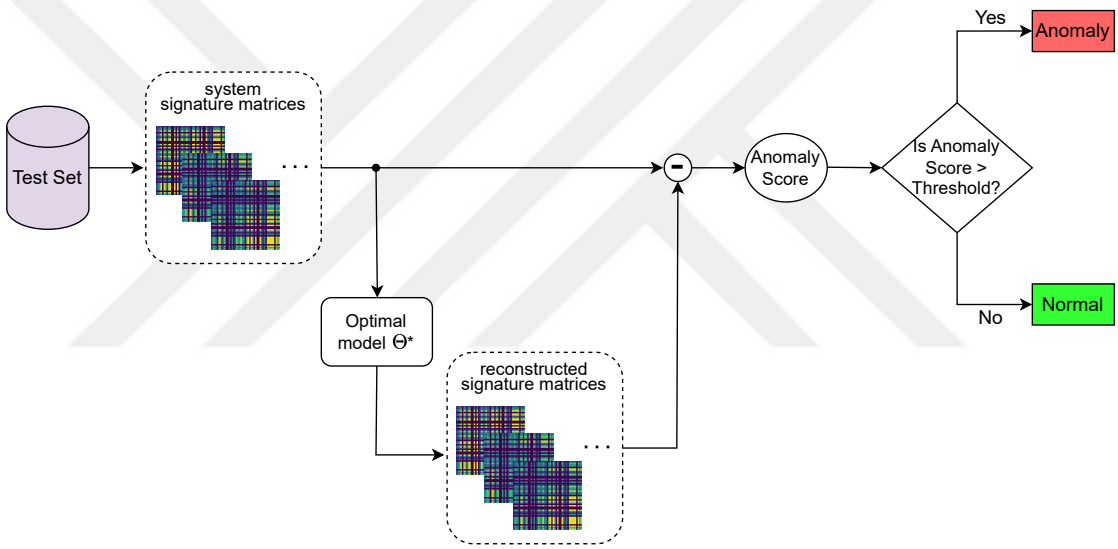

Figure 4.10. Reconstruction-based multivariate time series anomaly detection.

4.5. Experiments

In this section, comprehensive experiments are conducted using different data features as the difficulty measures for the data-based CL methods. A description of how the data is preprocessed before being used is also provided. The experiments are performed with the aim of addressing the following research questions (RQ):

(i) How can a training scheduler be designed for CL-based MVTS anomaly detection (RQ4.1)?

(ii) What is the most important data feature in MVTS that can enhance the perfor-

mance of CL-based models (RQ4.2)?

(iii) For data-based CL, is there always significant improvement in model performance for MVTS anomaly detection (RQ4.3)?

(iv) Does extending the CL concept to the learner result in improved anomaly detection performance (RQ4.4)?

4.5.1. Data Preprocessing

The original SWaT dataset consists of $946,722$ samples, which is a significantly large amount of data points and requires a lot of training time before conducting anomaly detection. To increase training efficiency, all the time series data in the SWaT dataset are first downsampled by a factor of ten. Also, the initial data points before the system attains stability are removed to prevent them from affecting the model performance. The resulting SWaT dataset used in this thesis consists of $49,500$ training data points and $44,992$ test points with $5,463$ anomalies which is approximately 12.14% of the test set. Downsampling the dataset by this factor does not significantly harm the performance of the algorithm. Instead, it provides computational advantages as this notably reduces the training time.

Next, the dataset is normalized using the maximum and minimum values of each time series data. The normalization process aims to enhance the learning capability and robustness of the model. Normalization also enhances data or pattern interpretability as the values from the sensors and actuators vary significantly. Data normalization is performed on both the training and test sets using the min-max normalization method expressed as

$$\widetilde{x} = -1 + \frac{2 \times (x - min(\mathbf{x}))}{max(\mathbf{x}) - min(\mathbf{x})} \tag{4.7}$$

where x represents the data point, $min(\mathbf{x})$ and $max(\mathbf{x})$ represent the minimum and maximum values of each time series, respectively. The normalized data denoted as $\widetilde{x}$ and bounded within the interval $[-1, 1]$ is used to generate the SSM for both model training and testing.

4.5.2. Model Architecture

The SWaT dataset contains 51 time series data from various sensors and actuators. For training and testing, 51 × 51 SSM to represent each row in the dataset are generated. In MSCRED, each of the data layers is represented as a three-dimensional array with dimensions of $n \times n \times d$, where $n \times n$ represents spatial dimensions (width × height) and d is the channel dimension (depth). The training utilizes an MSCRED architecture comprising 4 convolutional encoders (Conv1 to Conv4). These encoders consist of 32 kernels of size $1 \times 1 \times 3$, 64 kernels of size $1 \times 1 \times 32$, 128 kernels of size $1 \times 1 \times 64$, and 256 kernels of size $1 \times 1 \times 128$ respectively. All convolutional encoder layers have a stride of 1×1. A step size of 2 is chosen for the attention-based ConvLSTM layer. To reconstruct the SSM, 4 convolutional decoders (DeConv4 to DeConv1) are employed. These decoders consist of 128 kernels of size $1 \times 1 \times 256$, 64 kernels of size $1 \times 1 \times 128$, 32 kernels of size $1 \times 1 \times 64$, and 3 kernels of size $1 \times 1 \times 64$, each with a stride size of 1×1.

Extensive preliminary experiments are conducted to determine the best learning rate to be used for the proposed designs. This was done by varying the learning rates and evaluating the MSCRED model performance at different epochs as shown in Table 4.1 and Figure 4.11. These results indicate that the largest learning rate which is 10^{-3} in these experiments, gives the worst anomaly detection performance in the earlier iterations. The model was evaluated at 5 training epochs and the learning rate of 10^{-6} gives the best performance. After 10 and 15 epochs, 10^{-4} offers significantly better results than 10^{-3} albeit slightly better than and 10^{-5} and 10^{-6}. Hence, a learning rate of 10^{-4} was chosen for the proposed designs. After 20 epochs, apart from 10^{-3}, the performance with other learning rates starts diminishing which shows the importance of the learning rate and number of epochs. All the models including the proposed frameworks are implemented on PyTorch with the Adam optimizer.

Table 4.1. Anomaly detection results on SWaT dataset using MSCRED at various learning rates and epochs.

Learning Rate	**Epochs**	**Precision (%)**	**Recall (%)**	**F1 (%)**
0.000001	5	92.39	67.08	77.73
0.000001	10	96.23	67.83	79.57
0.000001	15	97.61	66.73	79.27
0.000001	20	97.04	67.08	79.32
0.000001	25	92.40	67.14	77.77
0.00001	5	55.48	69.05	61.52
0.00001	10	91.08	71.03	79.82
0.00001	15	91.05	71.08	79.84
0.00001	20	90.93	70.42	79.37
0.00001	25	90.98	67.42	77.45
0.0001	5	76.43	61.90	68.40
0.0001	10	95.00	68.71	79.75
0.0001	15	91.48	71.03	79.97
0.0001	20	91.82	66.60	77.20
0.0001	25	71.34	70.76	71.05
0.001	5	53.66	26.52	35.49
0.001	10	41.68	25.77	31.84
0.001	15	28.78	40.97	33.81
0.001	20	89.15	71.10	79.11
0.001	25	89.67	71.03	79.27

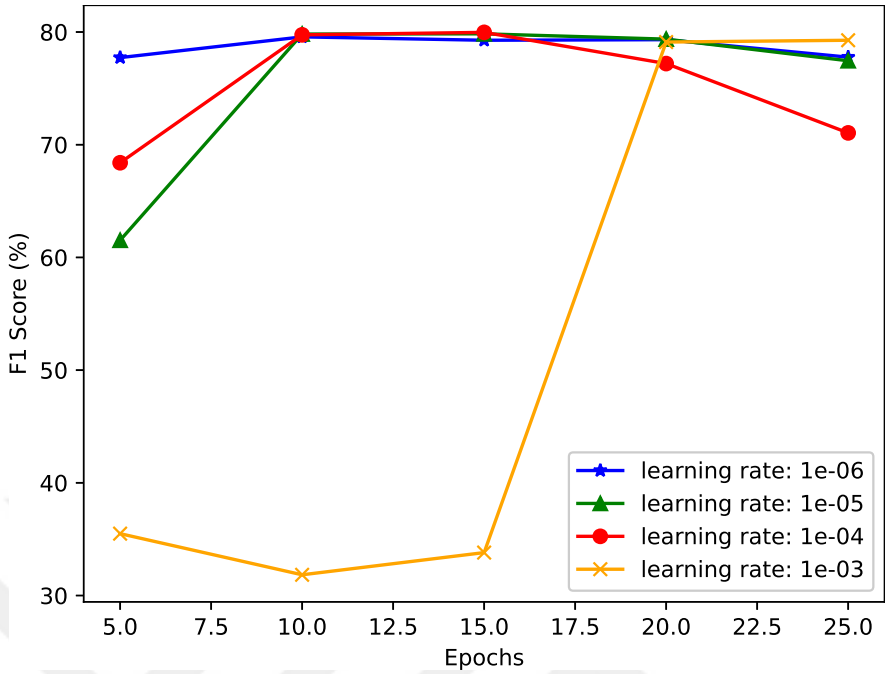

Figure 4.11. F1 scores of the MSCRED model at various learning rates and epochs using Adam optimizer.

4.5.3. Training Scheduler Design

As explained in Chapter 3, a training scheduler is required for the proposed CL-based frameworks. To do so, two schedulers shown in Figure 4.12 are considered: training for 15 epochs at each round and training with a total of 15 epochs for all the rounds combined (RQ4.1). Then the experiments were conducted using the proposed DCL methods based on sequence length and window size.

From the results presented in Table 4.2 and Table 4.3, it can be seen that utilizing 15 epochs per training round does not enhance model performance. Also, the training duration was significantly prolonged. Therefore, $E = 15$ epochs designated as the total number of epochs for each proposed design was selected for all the CL-based experiments. For curriculum designs comprising two training rounds, experiments in the initial round (with smaller data) were performed with 5 epochs, and 10 epochs were selected for the final training round. For instance, in DCL-MSCRED-P2, the first training round with the shorter data length was performed with 5 epochs, followed by a

10-epoch training round with the longer sequence, which is the whole data. For designs involving three difficulty levels, each training round was conducted for 5 epochs.

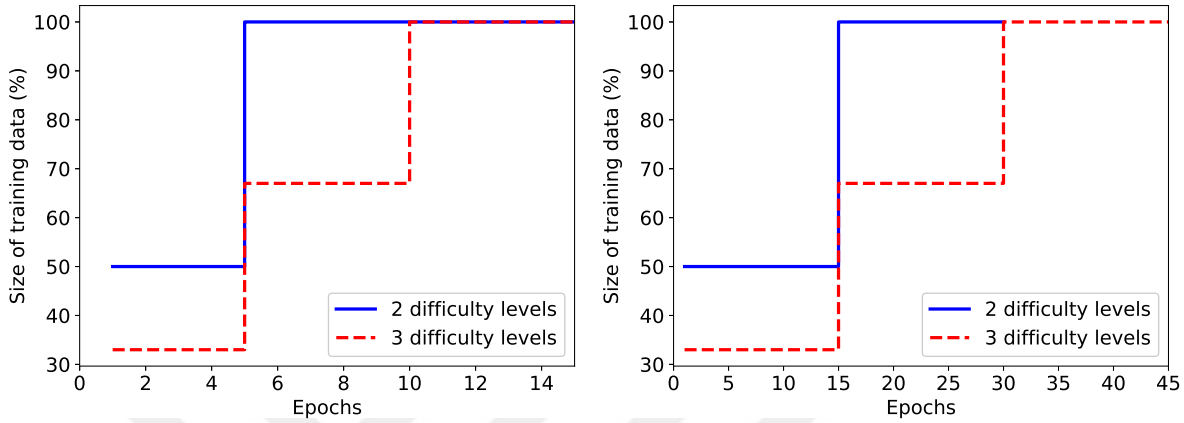

Figure 4.12. (Left) Training scheduler with a total of 15 epochs. (Right) Training scheduler with 15 epochs per training round.

Table 4.2. Anomaly detection results on SWaT dataset with two difficulty levels using different training schedulers. * represents training with easy subset for 5 epochs and then with whole training set for 10 epochs. ** represents training for 15 epochs at each training round.

Model	**Precision (%)**	**Recall (%)**	**F1 (%)**
DCL-MSCRED-P2 *	98.83	69.19	81.40
DCL-MSCRED-P2 **	94.46	69.73	80.24
DCL-MSCRED-W2 *	95.10	71.37	81.54
DCL-MSCRED-W2 **	94.59	69.12	79.87

Table 4.3. Anomaly detection results on SWaT dataset with three difficulty levels using different training schedulers. * represents training with easy subset for 5 epochs at each training round. ** represents training for 15 epochs at each training round.

Model	**Precision (%)**	**Recall (%)**	**F1 (%)**
DCL-MSCRED-P3 *	97.25	70.01	81.41
DCL-MSCRED-P3 **	93.97	69.05	79.61
DCL-MSCRED-W3 *	96.07	71.64	82.08
DCL-MSCRED-W3 **	97.72	70.21	81.71

4.5.4. Anomaly Detection Results and Discussion

Table 4.4. Anomaly detection results of the DCL-MSCRED-W3 model on SWaT dataset at different reconstruction error threshold values.

θ_R $(\times 10^{-6})$	**Precision (%)**	**Recall (%)**	**F1 (%)**
0.2	93.62	70.07	80.16
0.4	94.91	70.85	81.13
0.6	96.07	71.64	82.08
0.8	93.44	69.94	80.00
1.0	87.1	69.94	77.58
1.2	83.01	69.26	75.51
1.4	84.49	68.71	75.59
1.6	86.38	67.42	75.73
1.8	87.1	67.21	75.88
2.0	85.29	67.21	75.18

This section presents anomaly detection results of the DCL-MSCRED-W3 model at various θ_R values in Table 4.4 and the proposed approaches in Table 4.5. The results of the proposed methods also include a model trained with anti-CL, that is, starting from hard to easy. A comparison of the best CL design with the state-of-the-art

techniques is presented in Table 4.6.

Table 4.5. Anomaly detection results of designed curriculum learning-based methods on SWaT dataset.

Model	**Precision (%)**	**Recall (%)**	**F1 (%)**
DCL-MSCRED-P2	98.83	69.19	81.40
DCL-MSCRED-P3	97.25	70.01	81.41
DCL-MSCRED-W2	95.10	71.37	81.54
DCL-MSCRED-W3	96.07	71.64	82.08
DCL-MSCRED-S2	94.24	70.21	80.47
DCL-MSCRED-S3	97.80	69.67	81.37
DCL-MSCRED-N2	94.81	69.11	79.95
DCL-MSCRED-N3	94.34	69.33	79.92
MCLAD	96.25	68.23	79.86
D-MCLAD	94.80	70.82	81.08
Anti-MCLAD	89.59	62.78	73.83

Table 4.4 shows how the best F1 score was obtained for the DCL-MSCRED-W3 model by considering several θ_R values. The same evaluation strategy is used for the rest of the models to maximize the F1 score. Generally, utilizing CL for model training results in improved model performance. Among the proposed designs, DCL-MSCRED-W3 provides a 2.17% increase in F1 score when $\theta_R = 0.6 \times 10^{-6}$ in comparison to MSCRED, indicating the most improvement in anomaly detection performance. This can be credited to employing window size as the difficulty measure (RQ4.2). By initially capturing short local patterns, the model manages to converge faster and benefits from improved learning dynamics. This enhances optimization for subsequent training rounds as the model has more optimal initial values compared to standard training with the whole dataset. Gradually introducing medium and long-term patterns exposes the model to a variety of data which in turn aids generalization, thus, further improving performance.

Among the data-based designs, DCL-MSCRED-N3 provides the most insignificant increase in the F1 score. This shows that reducing the noise at the initial training rounds does not have a significant impact on model learning which indicates that CL does not always result in significant performance improvement (RQ4.3). The diversity of the results from the data-based frameworks, ranging from 79.92% to 82.08% shows that there exists a data feature in MVTS that is the most significant for optimal results. In this thesis, the window size proves to be the most important feature for CL-based designs. These time series features can further be combined to obtain several data-based CL approaches.

The model-based CL design, MCLAD, yields the worst F1 score among the implemented CL designs. This suggests that training a less complex LSTM-ED model with difficult data results in a substandard initial model to be shared for the subsequent training round (RQ4.4). On the other hand, the performance of D-MCLAD underpins the significance of a CL design for anomaly detection. Integrating data-based CL with the model-based framework, a 1.22% F1 score improvement is observed for D-MCLAD in comparison to MCLAD. This indicates that while a model-based CL design alone may not provide performance improvement, training a less complex model with easy data can facilitate transferring better model parameters to the subsequent training round resulting in better performance.

Moreover, to further justify the benefits of easy-to-hard ordered learning, an anti-CL strategy is used in MCLAD. In Anti-MCLAD, the more complex MSCRED model is initially trained with the entire training set and then its parameters are transferred to be the initial parameters of the less complex LSTM-ED model using the whole dataset as well. The evaluation is done after training with the LSTM-ED and the results show a significant performance decrease compared to both MCLAD and MSCRED indicating the importance of a training curriculum.

4.5.5. Comparison with State-of-the-Art Models

The anomaly detection results of the best performing DCL-MSCRED-W3 is compared with the state-of-the-art models using SWaT dataset as presented in Table 4.6.

Table 4.6. Anomaly detection results of the best curriculum learning-based design and the state-of-the-art techniques on SWaT dataset.

Model	**Precision (%)**	**Recall (%)**	**F1 (%)**
DAGMM	27.46	69.52	39.00
AE	72.63	52.63	61.00
RLMSAD	81.05	60.87	69.45
LSTM-ED	78.60	66.60	72.10
LSTM-VAE	96.24	59.91	74.01
SUSAN	63.30	91.00	74.70
MSCRED	94.74	69.09	79.91
GAFM	89.10	74.34	81.06
DCL-MSCRED-W3	96.07	71.64	82.08

The autoencoder-based (AE) [65] framework exhibits superior performance compared to the Deep Autoencoding Gaussian Mixture Model (DAGMM) [66], primarily due to its capability to capture non-linear correlations within the time series data. In DAGMM, anomaly detection was carried out without taking the temporal dependencies in the data into account. Instead of feeding the model with temporal patterns, it was provided with a single observation containing many features. Since temporal pattern learning is important for MVTS anomaly detection, DAGMM produced the least F1 score.

LSTM-based frameworks such as LSTM-ED [64] and LSTM-VAE [67] outperform AE and a reinforcement learning-based RLMSAD [68] framework owing to the adaptive learning capacity of the LSTM, enabling it to focus on relevant information while

forgetting less pertinent features during training. MSCRED combines LSTM with CNNs, incorporating an attention mechanism to enhance anomaly detection performance by effectively capturing temporal patterns. Notably, the attention mechanism proves adept at learning only the most relevant information essential for learning temporal patterns. SUSAN [35] was designed to maximize the recall even if it reduces the precision, however, permitting the model to have high false positive classifications results in a non-optimal F1 score.

A time series encoding framework based on Gramian Angular Field Matrix (GAFM) [69], which performs better than MSCRED is considered. Gramian Angular Field is a technique for embedding time series, establishing a one-to-one mapping between time series data and 2D matrices. However, by introducing CL to the MSCRED framework, some of the implemented data-based CL frameworks and D-MCLAD outperform GAFM.

4.6. Summary of the Chapter

In this chapter, the proposed CL-based approaches for training a deep learning framework were presented. First, a description of the water treatment dataset and examples of the targeted anomalies were provided. Then DCL approaches that consider MVTS features such as sequence length, window size, time series stationarity, and noise level were proposed. Subsequently, the efficacy of a model-based CL framework was analyzed. Then these two approaches were combined to further study the importance of the curriculum design in model performance. Preliminary experiments were then conducted to determine the best learning rate and to design a training scheduler. Finally, extensive experiments were conducted and the best CL-based design was compared with some of the related state-of-the-art techniques. This method is used in Chapter 5 for performing anomaly detection in real-world textile processes.

5. ANOMALY DETECTION IN TEXTILE PROCESSES

In textile industries, machines and processes are subject to a multiplicity of variables, thus, to guarantee high-quality products, process control techniques have to be utilized. The operation of textile dyeing machines can be affected by several factors that can lead to a temperature profile that is distinctly different from the expected one which can lead to substandard products. Anomaly detection for textile processes as presented in this chapter can help alleviate such problems.

5.1. Textile Processes

The textile industry was among the first dominant industries of the industrial revolution and the automatic control of dyeing machines emerged back in the 1960s [70]. Textile processes are various techniques and operations used in the large-scale production of textiles. These processes include fiber production, spinning, weaving, knitting, and chemical processing. Fiber production, one of the fundamental processes, involves the conversion of raw materials such as cotton, wool, and synthetic fibers such as polyester or nylon into thin, elongated strands used to produce the final textiles. Spinning involves twisting the fibers together using spinning machines such as ring frames, rotors, and air-jet spinning machines to create a continuous strand of yarn. Weaving involves interlacing the yarns in a warp and weft pattern to create a flat fabric using looms. Knitting involves interlocking the yarns in a series of loops to create a three-dimensional fabric using knitting machines. After the fabrics have been produced, they are typically subjected to a number of chemical processes such as dyeing to enhance their aesthetic and functional properties.

Dyeing involves applying dyes to the fabrics to give them color using dyeing machines which are embedded with control mechanisms to monitor the temperature, pH, and dyebath flow rate. Traditional dyehouses employ manual control techniques, therefore, an engineer or technician is always needed to monitor the dyeing process. With

the advancement in technology, microprocessors are being used to automate control operations but only a few of their features are utilized in this setup. In order to enhance the use of the capabilities of the microprocessor systems, textile machines are linked to a computer via CIM. This technique has changed the paradigm of process control in the textile industry. The main advantages of automation are quality improvement, cost-effectiveness, and easier manufacturing. In CIM systems, there is constant communication between the computer and machines. As such, real-time information of the ongoing processes can be easily accessed from the computer which significantly enhances the decision-making process.

ICS is an inherently challenging area for the implementation of anomaly detection systems due to the complexity of proprietary components. Textile dyeing machines are implemented with several of these components, hence, the causes of anomalies during the operation may be difficult to pinpoint.

5.1.1. Control of Dyeing Processes

The aim of conventional dyeing process control techniques is to get the desired outcomes by ensuring minimum deviation from the specified process profile such as temperature. In some cases, outcomes with uncontrollable deviations from the standard process profile may be accepted and then corrected later. Contrary to traditional techniques which are process-oriented, recent methodologies are predictive and result-oriented. These approaches use real-time data to constantly forecast the desired outcome.

In [70], several process control techniques are presented. One of these techniques aims at monitoring and tuning controllable variables to achieve the desired product property such as dye shade. The dyeing processes may not consistently have the same parameters, for instance, there may be variations in water quality, substrate preparation, or raw ingredients but they all have the same desired result. In traditional control techniques, if the process profile deviates from the standard one, then the

controller will adjust some parameters to minimize the deviation. For example, if the temperature at a particular time deviates from the standard profile, the controller will activate the steam valves to regulate the temperature. This proposed method does not regulate the temperature, but rather forecasts on the impact of the deviation on the desired product. The controller will undertake the optimal action such as changing the pH of the dyebath if the variation leads to an unfavorable dye shade but will not take any action if the desired dye shade can still be obtained. High-quality products may be achieved by integrating real-time, predictive, and adaptive process control ideas with cutting-edge computer software, hardware, and monitoring sensors. Furthermore, using these techniques, anomalies that can affect the quality of the products can be detected. Some of the temperature-related anomalies are given below:

- Constant temperature: The temperature stays constant even though an input, either heating or cooling is applied. This anomaly may be caused by communication problems or physical restrictions in the valves.
- Incorrect temperature readings: Temperature readings may be out of sensor range or unrealistic due to filtering, calibration, or sensor-related problems.
- Rapid temperature changes: Sometimes the temperature changes very rapidly even though a small amount of input is applied. This may be due to the wrong configuration of some related parameters.
- Self-increase in temperature: The temperature may increase even though heating is not applied. This may be due to factors like steam leakage.
- Temperature oscillations: It is undesirable to have oscillations because not only do they increase consumption, but they may also reduce dyeing quality.
- Noisy temperature readings: Temperature readings can be noisy due to PLC or sensor-related problems.
- Unnecessary cooling: This may cause excess consumption and deterioration in the process control.

5.1.2. Challenges in Anomaly Detection for Textile Processes

The integration of anomaly detection systems with industrial machines can help prevent equipment malfunction and security threats to ensure efficient system operation. Nevertheless, there are several challenges that may arise when implementing anomaly detection algorithms in ICS. Also, the choice of the anomaly detection technique comes with its own benefits and drawbacks. Below are some of the main challenges in anomaly detection for textile processes:

- Anomaly detection algorithms rely on the quality and abundance of data for training. Textile processes are generally complex and dynamic, hence, the data generated may be noisy or incomplete which may negatively impact the anomaly detection accuracy.
- Anomaly detection techniques may falsely detect anomalies or fail to identify them. False alarms may cause unnecessary disruptions to the system which can result in increased costs and reduced efficiency. On the other hand, missed detections may cause equipment failure or increase vulnerability in security.
- In most cases, it is required to detect anomalies in textile processes in real time. This requires the use of efficient and scalable anomaly detection algorithms that can accurately process data in real time. Developing such frameworks that precisely fit into the existing workflow is a significantly challenging task.
- The textile industry involves a wide variety of materials and processes, each of which may have unique characteristics that affect the performance of anomaly detection algorithms. Therefore, it may be difficult to develop a universal anomaly detection mechanism that is effective across all textile processes.
- The performance of anomaly detection techniques may be impacted by the decisions of the engineers or technicians monitoring the processes. For example, they may override the recommendations of the anomaly detection algorithms, which can affect the accuracy of the algorithms.
- Textile machinery employ a wide range of sensors to generate large amounts of data in real time which may be difficult to analyze. The high dimensionality

of data and imbalance between normal and anomalous data points necessitates the need not only for high-performance techniques, but also for fast anomaly detection algorithms.

- The textile industry has several interconnected components and processes that interact and influence each other in complex ways making it difficult to accurately distinguish normal system behavior from anomalous one.
- Textile processes are often subject to external factors such as temperature or pressure. These factors can affect the normal operation of the system, hence, it may be difficult to precisely identify what constitutes an anomaly. For example, a sudden increase in temperature may still produce the desired output, but this increase may be caused by malfunctioning equipment.

5.2. Textile Dyebath Temperature Dataset

In this research, a textile dataset that contains two time series, which are reference temperature (T_{ref}) and the actual temperature (T_{act}) of dyebath is used. The training set shown in Figure 5.1 consists of $58,739$ data points collected on 11 April, 2023 from actual textile dyehouses where each data point represents one second.

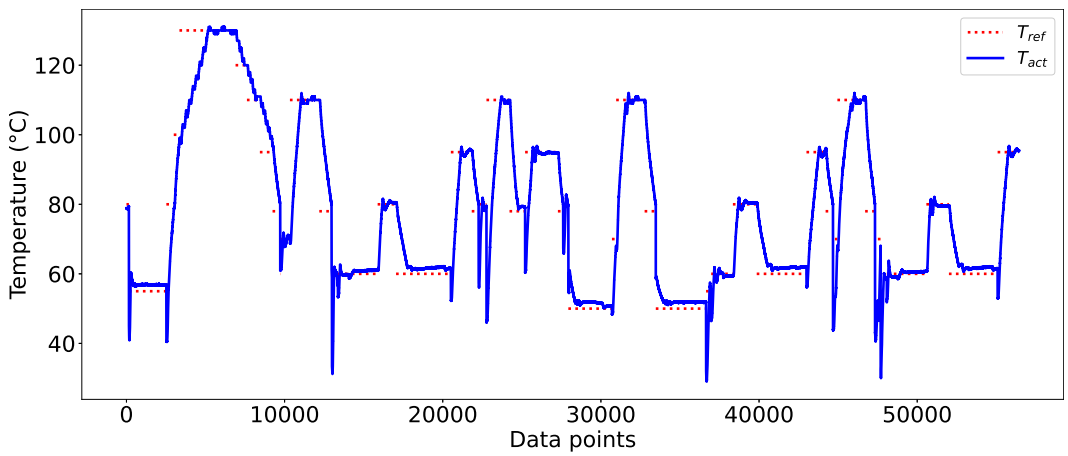

Figure 5.1. Original training set of the textile dyebath temperature dataset.

In this dataset, all dyeing processes occur in batches with temperatures fluctuating up to 130°C. Ideally, as dyeing progresses, the dyebath temperature should follow the reference. Therefore, standard operational processes should display patterns that either closely track the reference or show minimum tracking error. A normal batch temperature pattern resembles a damped oscillation, that is, the tracking error decreases as the dyeing process continues. Also, the temperature that converges slightly above or below the reference is considered normal. Therefore, any batch temperature showing undamped or non-decaying oscillations and noisy readings is considered an anomaly. Concisely, in this thesis the following anomalies are targeted: temperature oscillations (A_1) and noisy temperature readings (A_2). The training set does not contain any anomaly but all the test sets consist of at least one collective anomaly. Table 5.1 provides the summary of the test sets, including the anomalies present in each set.

Table 5.1. Description of the test sets of the textile dyebath temperature dataset.

Anomaly type	**Test set size**	**Anomaly batches**	**Anomaly size**
A_1	16,142	1	1,000
A_2	16,141	1	999
A_1 and A_2	17,142	2	1,999

5.3. Experiments

In this section, several experiments were conducted using the best performing DCL-MSCRED-W3 model designed in Chapter 4. The aim is to perform collective anomaly detection, hence, window sizes $\omega = \{60, 90, 120\}$ are used to represent short, medium, and long windows, respectively. Since the textile dataset consists of two time series, the correlation matrices are of size 2×2. The rest of the model architecture, training scheduler, and hyperparameters are the same as in the DCL-MSCRED-W3 model. A description of how the data is preprocessed before being used is also provided. The experiments were conducted with the aim of addressing the following research questions:

(i) Can a deep learning model be used for anomaly detection in real-world textile processes (RQ5.1)?

(ii) What is the effect of a reconstruction error threshold in the model performance (RQ5.2)?

(iii) Is the model performance affected by the presence of different types of collective anomalies (RQ5.3)?

5.3.1. Data Preprocessing

The data collected from the dyeing machine consists of several controlled and uncontrolled regions. Anomaly detection cannot be applied to uncontrolled regions as there is no predefined expected behavior, hence, all uncontrolled regions are removed. The resulting dataset shown in Figure 5.1 consists of several controlled batches. However, as system statuses cannot change instantaneously, the real-time dyeing temperature takes time to stabilize. These data points are not useful for model training and can negatively affect model performance. Therefore, they are removed by considering the difference between the reference and the dyeing temperature. If the difference is greater than the desired threshold, 2.5°C in this work, they are removed to obtain a dataset shown in Figure 5.2 which consists of 34, 085 data points.

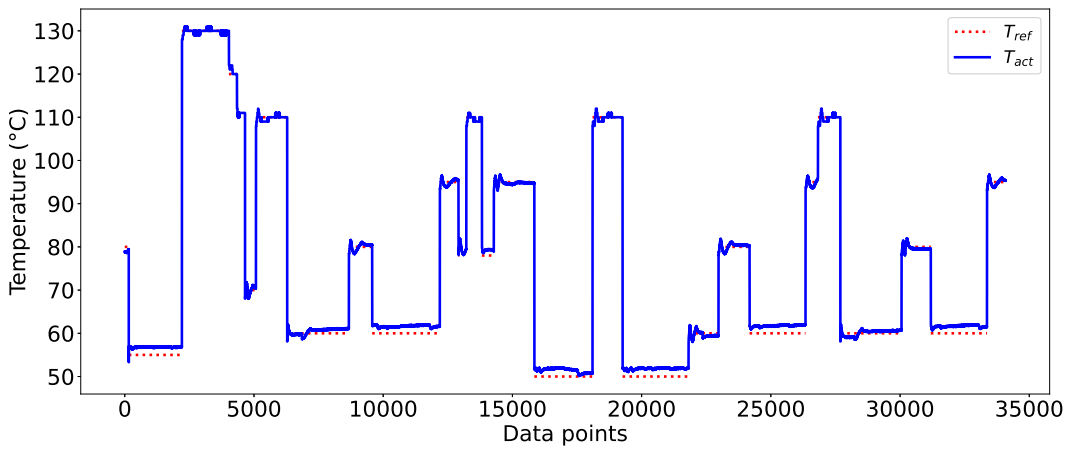

Figure 5.2. Cleaned training set of the textile dyebath temperature dataset.

As seen in Figure 5.2, there is a large jump or drop in temperature as the reference changes from one batch to another. This may be learned as part of the temporal patterns, resulting in poor anomaly detection performance. Therefore, the reference value is shifted to 50°C throughout the dataset while maintaining the patterns and differences between the references and temperature readings the same. Also, the training data is thoroughly inspected such that it comprises no anomaly. The anomalies are only in the test sets as shown from Figure 5.3 to 5.5.

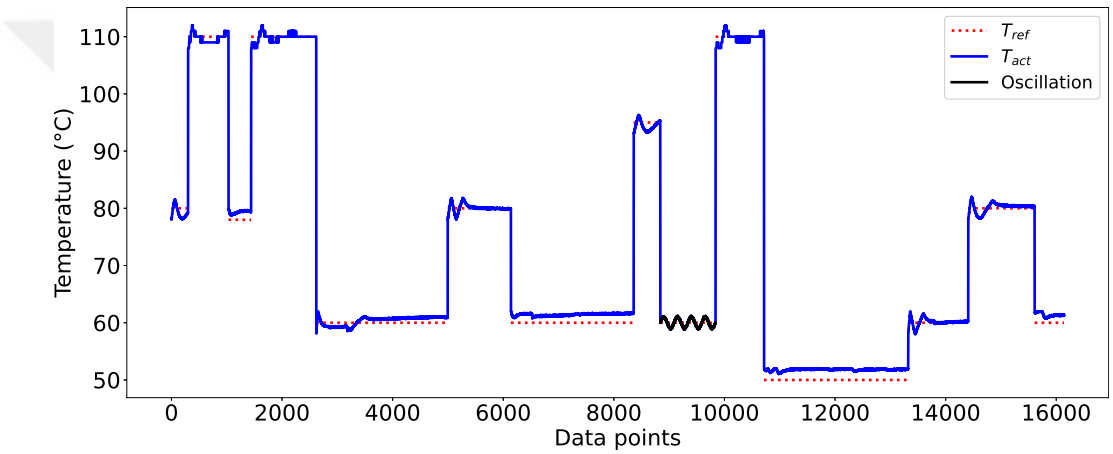

Figure 5.3. Test set of the textile dyebath temperature dataset with an oscillation anomaly.

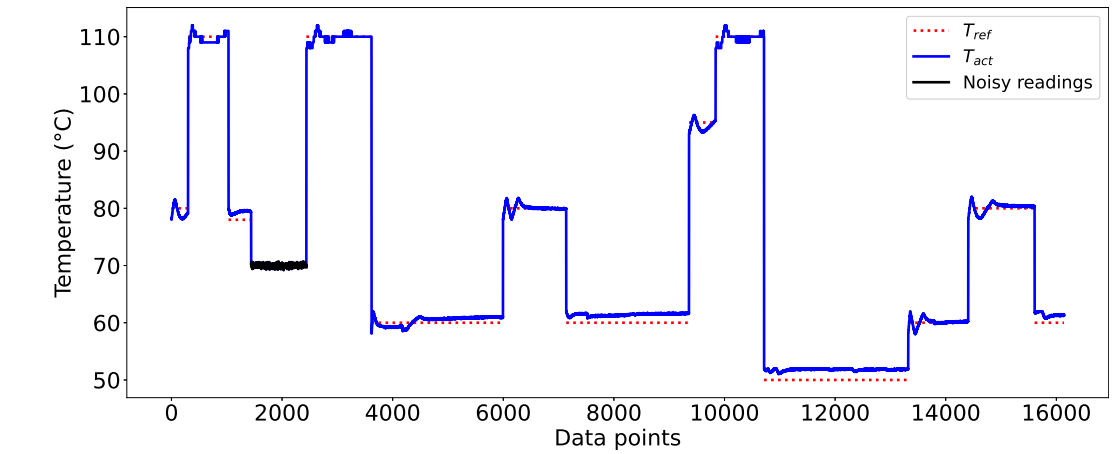

Figure 5.4. Test set of the textile dyebath temperature dataset with noisy temperature readings.

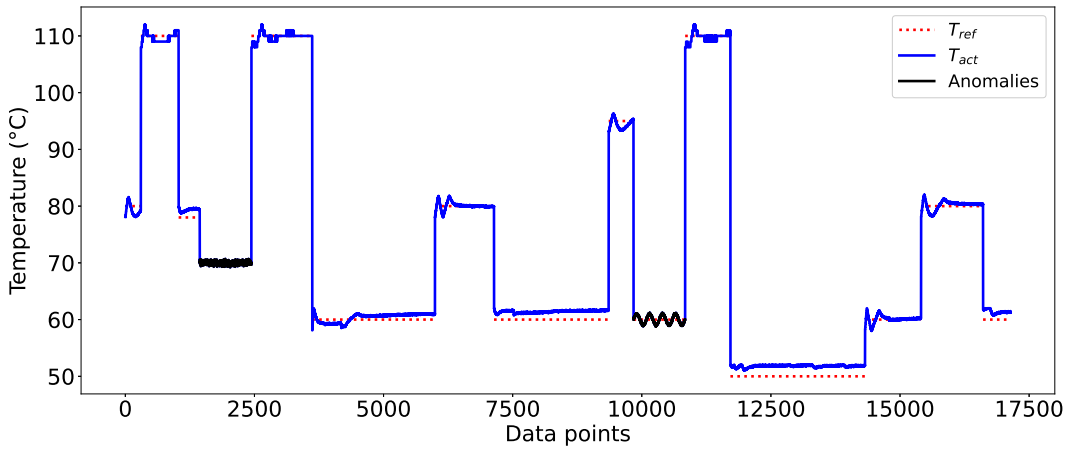

Figure 5.5. Test set of the textile dyebath temperature dataset with an oscillation anomaly and noisy temperature readings.

5.3.2. Anomaly Detection Results and Discussion

This section provides the collective anomaly detection results showing the anomaly scores which are the average reconstruction errors of each batch. The figures presented in this section consist of the left y-axis which shows the temperature, the right y-axis which shows the anomaly scores obtained at different reconstruction error thresholds, and the x-axis which represents the time stamps of the data in seconds.

In this chapter, the proposed DCL-MSCRED-W3 model is used to classify anomalous batches from normal ones in the textile dataset. This approach is chosen because experimental results in Chapter 4 show that the window size is the most effective feature for improved performance among the studied features. Also, since the textile dataset consists of only two time series, CL approaches that feed the deep learning model with a fewer number of time series such as those based on time series stationarity or noise level are not applicable. This is because correlation matrices can only be obtained from MVTS data while such approaches necessitate the use of only one time series for textile dataset during the easy training round.

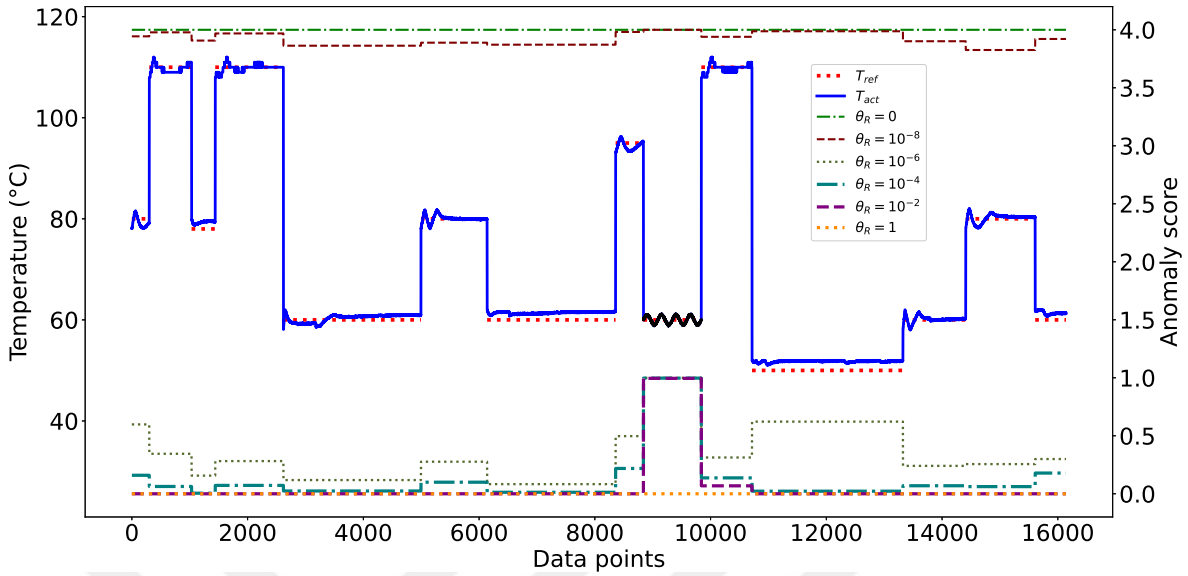

Figure 5.6. Collective anomaly detection on a test set with an oscillation anomaly at various reconstruction error thresholds.

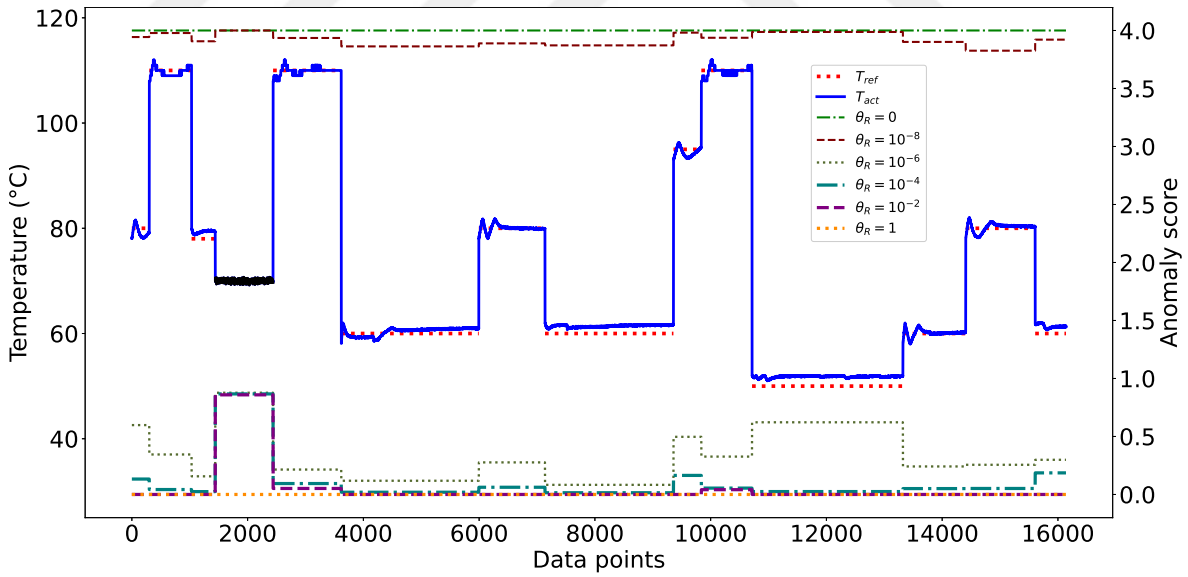

Figure 5.7. Collective anomaly detection on a test set with noisy temperature readings at various reconstruction error thresholds.

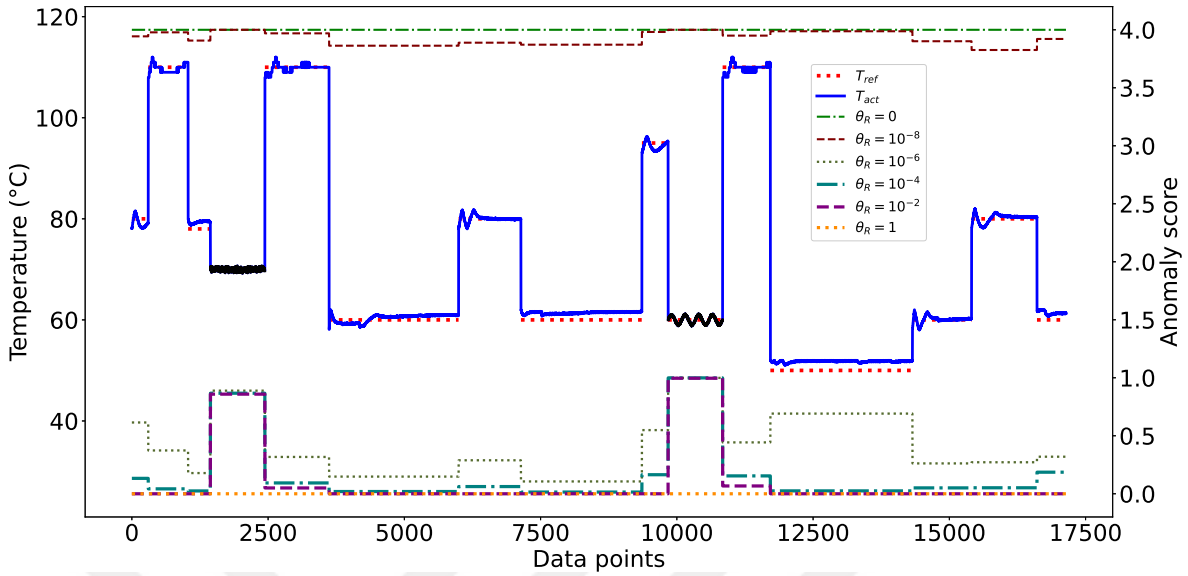

Figure 5.8. Collective anomaly detection on a test set with an oscillation anomaly and noisy temperature readings at various reconstruction error thresholds.

Since the dyeing temperature readings are controlled in batches, collective anomaly detection is conducted with the aim of perfectly classifying anomalous batches from normal patterns. As shown in Figures 5.6 to 5.8, the model performs successful classification in real-world textile data at non cut-off thresholds $\theta_R = 0$ and $\theta_R = 1$, which answers RQ5.1. These figures show that the anomaly scores of the previously seen normal patterns during training are lower than those of the unseen noisy readings and oscillation batches at threshold values from $\theta_R = 10^{-8}$ to 10^{-2}.

The first evaluation is performed on a dataset that consists of just an oscillation anomaly as shown in Figure 5.6. When $\theta_R = 0$, the model classifies all batches as anomalies, and when $\theta_R = 1$, the anomaly score is 0 for all batches indicating that no anomaly is detected. As θ_R increases, the capability of the model to classify between normal and anomalous batches improves because it outputs high anomaly scores for the oscillation anomaly and relatively low scores for normal batches. It is observed that as the reconstruction error threshold increases from $\theta_R = 10^{-8}$ to 10^{-2}, the anomaly scores of normal batches decrease from a maximum of about 3.95 to less than 0.1, respectively, and the anomaly score of the oscillating readings remains the highest (RQ5.2).

The second evaluation is conducted on a dataset that consists of only noisy temperature readings as shown in Figure 5.7. It can be seen that the batch with noisy readings has an anomaly score of about 4 compared to less than 3.95 for the normal batches when $\theta_R = 10^{-8}$. Although the anomaly batch has the highest anomaly score, the scores of some normal batches are very close to 4. This may not result in robust anomaly detection which shows the importance of a reconstruction error threshold. Similar to oscillation anomalies, as the threshold increases, the anomaly scores of noisy readings remain the highest whereas those of normal batches decrease from a maximum of about 3.95 at $\theta_R = 10^{-8}$ to almost 0 at $\theta_R = 10^{-2}$.

Lastly, the model is evaluated on a test set consisting of both the temperature oscillations and noisy readings. Similarly, the evaluation is conducted at different reconstruction thresholds and the model successfully detects these anomalies at $\theta_R = 10^{-8}$ to 10^{-2} as shown in Figure 5.8 which addresses RQ5.3. Overall, $\theta_R = 10^{-2}$ offers the most significant difference in anomaly scores between normal and anomalous batches. Figure 5.8 also shows that the model detects the oscillation anomalies with a higher anomaly score than in the noisy readings. This can be attributed to the significant and more observable fluctuations of the oscillations compared to the local fluctuations of the noisy temperature readings.

5.4. Summary of the Chapter

This chapter presented anomaly detection in real-world textile processes. The aim was to investigate the potential of deep learning to detect anomalies not only in benchmark datasets but also in actual industrial data. First, different types of textile processes such as fiber production and dyeing were explained. Then a control technique for dyeing processes and types of anomalies were also presented, Next, the challenges of anomaly detection in textile processes were provided. The CL-based framework proposed in Chapter 4 was used to conduct experiments and analyze the severity of the noisy temperature readings and oscillation batch anomalies.

6. CONCLUSION

In this thesis, MVTS anomaly detection was investigated in two different industrial processes. First, data-based and model-based CL approaches were proposed and evaluated on a benchmark water treatment dataset. In the data-based method, the significance of different data features such as window size and time series stationarity were analyzed. For the model-based approach, a simple LSTM-ED model was trained first and then a complex MSCRED model was employed in the final training round. The evaluation results indicate that data-based designs generally provide performance improvement, while the model-based design does not enhance the baseline frameworks. Consequently, the data-based approach was combined with the model-based design, resulting in improved anomaly detection performance.

In the second part, a comprehensive analysis of collective anomaly detection in textile processes, an area that has received limited attention in this research area was performed. In this study, real-world textile data was collected, preprocessed, and used with the aim of determining whether deep learning anomaly detection frameworks can have practical applications. The dataset consists of noisy readings and oscillation anomalies which were targeted. Through extensive experiments with the CL method based on window size, it was concluded that deep learning can aid anomaly detection in the textile industry.

A potential future research direction would be investigating different MVTS encoding techniques. Also, since CL cannot be generalized across different fields, these methodologies can be extended to diverse case studies. For textile processes, future research can involve analyzing SPL, an automatic CL technique, by training the model with easy batches first and introducing difficult ones later.

REFERENCES

1. Dube, D. C. and M. Akar, "Curriculum Learning-Based Multivariate Time Series Anomaly Detection", *35th Conference of Open Innovations Association (FRUCT)*, pp. 186–193, Tampere, Finland, 2024.

2. Dube, D. C., M. Çom and M. Akar, "Temporal Pattern-Based Collective Anomaly Detection in Textile Processes", *Turkish Automatic Control Conference (TOK)*, Vol. 24, pp. 844–848, Istanbul, Turkey, 2023.

3. Wei, W. W., *Multivariate Time Series Analysis and Applications*, John Wiley & Sons, New Jersey, 2018.

4. Scheer, A., *CIM Computer Integrated Manufacturing: Towards the Factory of the Future*, Springer Science & Business Media, Berlin, 2012.

5. Wressnegger, C., A. Kellner and K. Rieck, "Zoe: Content-Based Anomaly Detection for Industrial Control Systems", *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 127–138, Luxembourg City, Luxembourg, 2018.

6. Thill, M., W. Konen, H. Wang and T. Bäck, "Temporal Convolutional Autoencoder for Unsupervised Anomaly Detection in Time Series", *Applied Soft Computing*, Vol. 112, pp. 107751–107773, 2021.

7. Azer, M. A., S. M. El-Kassas and M. S. El-Soudani, "A Survey on Anomaly Detection Methods for Ad Hoc Networks", *Ubiquitous Computing and Communication Journal*, Vol. 2, No. 3, pp. 67–76, 2005.

8. Grieser, J., S. Trömel and C.-D. Schönwiese, "Statistical Time Series Decomposition into Significant Components and Application to European Temperature", *Theoretical and Applied Climatology*, Vol. 71, pp. 171–183, 2002.

9. Samariya, D. and A. Thakkar, "A Comprehensive Survey of Anomaly Detection Algorithms", *Annals of Data Science*, Vol. 10, No. 3, pp. 829–850, 2023.

10. Nassif, A. B., M. A. Talib, Q. Nasir and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review", *IEEE Access*, Vol. 9, pp. 78658–78700, 2021.

11. Amos, J., "Artemis: NASA Calls Off New Moon Rocket Launch", 2022, `https://www.bbc.com/news/science-environment-62715616`, accessed on November 14, 2022.

12. Smith, S., "Out of Gas: A Deep Dive into the Colonial Pipeline Cyberattack", 2022, `https://sk.sagepub.com/cases/out-of-gas-a-deep-dive-into-the-colonial-pipeline-cyberattack`, accessed on November 15, 2022.

13. Robles, F. and N. Perlroth, "Dangerous Stuff: Hackers Tried to Poison Water Supply of Florida Town", 2021, `https://www.nytimes.com/2021/02/08/us/oldsmar-florida-water-supply-hack.html`, accessed on November 14, 2022.

14. Gibbs, S., "Triton: Hackers Take Out Safety Systems in Watershed Attack on Energy Plant", 2017, `https://www.theguardian.com/technology/2017/dec/15/triton-hackers-ma-lware-attack-safety-systems-energy-plant`, accessed on November 15, 2022.

15. Bilton, N., "Nest Thermostat Glitch Leaves Users in the Cold", 2016, `https://www.nytimes.com/2016/01/14/fashion/nest-thermostat-glitch-battery-dies-software-freeze.html`, accessed on November 14, 2022.

16. Wingfield-Hayes, R., "Dreamliners: Boeing 787 Planes Grounded on Safety Fears", 2013, `https://www.bbc.com/news/business-21054089`, accessed on November 15, 2022.

17. Bronk, C. and E. Tikk-Ringas, "The Cyber Attack on Saudi Aramco", *Survival*,

Vol. 55, No. 2, pp. 81–96, 2013.

18. Bengio, Y., J. Louradour, R. Collobert and J. Weston, "Curriculum Learning", *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, Montreal, Canada, 2009.

19. Dube, D. C., Ç. E. Erdem and Ö. Korçak, "CL-FedFR: Curriculum Learning for Federated Face Recognition", *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, Vol. 2, pp. 845–852, Rome, Italy, 2024.

20. Zhang, L., Z. Mao, B. Xu, Q. Wang and Y. Zhang, "Review and Arrange: Curriculum Learning for Natural Language Understanding", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 29, pp. 3307–3320, 2021.

21. Wang, X., Y. Chen and W. Zhu, "A Survey on Curriculum Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 44, No. 9, pp. 4555–4576, 2021.

22. Koenecke, A. and A. Gajewar, "Curriculum Learning in Deep Neural Networks for Financial Forecasting", *Mining Data for Financial Applications*, pp. 16–31, Springer, Würzburg, Germany, 2020.

23. Aggarwal, C. C., *Outlier Analysis: Second Edition*, Springer Nature, Cham, 2016.

24. Kromkowski, P., S. Li, W. Zhao, B. Abraham, A. Osborne and D. E. Brown, "Evaluating Statistical Models for Network Traffic Anomaly Detection", *Systems and Information Engineering Design Symposium (SIEDS)*, pp. 1–6, Charlottesville, United States, 2019.

25. Grzesiek, A., R. Zimroz, P. Śliwiński, N. Gomolla and A. Wyłomańska, "Long Term Belt Conveyor Gearbox Temperature Data Analysis - Statistical Tests for Anomaly Detection", *Measurement*, Vol. 165, pp. 108124–108135, 2020.

26. Villa, T., T. Kam, R. K. Brayton and A. L. Sangiovanni-Vincentelli, *Synthesis of Finite State Machines: Logic Optimization*, Springer Science & Business Media, New York, 2012.

27. Pedroni, V. A., *Finite State Machines in Hardware: Theory and Design (with VHDL and SystemVerilog)*, MIT Press, London, 2013.

28. Cheng, L., K. Tian, D. D. Yao, L. Sha and R. A. Beyah, "Checking is Believing: Event-aware Program Anomaly Detection in Cyber-physical Systems", *IEEE Transactions on Dependable and Secure Computing*, Vol. 18, No. 2, pp. 825–842, 2019.

29. Zhu, Y., Y.-Q. Ni, H. Jin, D. Inaudi and I. Laory, "A Temperature-driven MPCA Method for Structural Anomaly Detection", *Engineering Structures*, Vol. 190, pp. 447–458, 2019.

30. Li, X., J. Li, A. Abdollahi and T. Jones, "Data-driven Thermal Anomaly Detection for Batteries using Unsupervised Shape Clustering", *30th IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 1–6, Kyoto, Japan, 2021.

31. Hutter, F., L. Kotthoff and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, Springer Nature, Cham, 2019.

32. Chatterjee, S., R. Bopardikar, M. Guerard, U. Thakore and X. Jiang, "MOSPAT: AutoML based Model Selection and Parameter Tuning for Time Series Anomaly Detection", ArXiv:2205.11755, 2022.

33. Wagner, D., T. Michels, F. C. Schulz, A. Nair, M. Rudolph and M. Kloft, "TimeSeAD: Benchmarking Deep Multivariate Time-Series Anomaly Detection", *Transactions on Machine Learning Research*, 2023.

34. Munir, M., S. A. Siddiqui, A. Dengel and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series", *IEEE Access*,

Vol. 7, pp. 1991–2005, 2018.

35. Gómez, Á. L. P., L. F. Maimó, A. H. Celdrán and F. J. G. Clemente, "SUSAN: A Deep Learning Based Anomaly Detection Framework for Sustainable Industry", *Sustainable Computing: Informatics and Systems*, Vol. 37, pp. 100842–100854, 2023.

36. Huong, T. T., T. P. Bac, K. N. Ha, N. V. Hoang, N. X. Hoang, N. T. Hung and K. P. Tran, "Federated Learning-Based Explainable Anomaly Detection for Industrial Control Systems", *IEEE Access*, Vol. 10, pp. 53854–53872, 2022.

37. Li, T., A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions", *IEEE Signal Processing Magazine*, Vol. 37, No. 3, pp. 50–60, 2020.

38. Lim, H., S. Park, M. Kim, J. Lee, S. Lim and N. Park, "MadSGM: Multivariate Anomaly Detection with Score-based Generative Models", *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1411–1420, New York, United States, 2023.

39. Mitchell, T. M., *Machine Learning*, McGraw Hill, New York, 1997.

40. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Massachusetts, 2016.

41. Zhang, A., Z. C. Lipton, M. Li and A. J. Smola, "Dive into Deep Learning", ArXiv:2106.11342, 2023.

42. Medsker, L. R., L. Jain *et al.*, "Recurrent Neural Networks", *Design and Applications*, Vol. 5, No. 64-67, p. 2, 2001.

43. Dudek, G., S. Smyl and P. Pełka, "Recurrent Neural Networks for Forecasting Time Series with Multiple Seasonality: A Comparative Study", *International Conference*

on Time Series and Forecasting, pp. 171–185, Gran Canaria, Spain, 2022.

44. Zhang, C., D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen and N. V. Chawla, "A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 1409–1416, Hawaii, United States, 2019.

45. Hallac, D., S. Vare, S. Boyd and J. Leskovec, "Toeplitz Inverse Covariance-based Clustering of Multivariate Time Series Data", *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 215–223, Halifax, Canada, 2017.

46. Shelhamer, E., J. Long and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation.", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 4, pp. 640–651, 2017.

47. Shi, X., Z. Chen, H. Wang, D. Yeung, W. Wong and W. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting", *Advances in Neural Information Processing Systems*, Vol. 28, 2015.

48. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", ArXiv:1412.6980, 2014.

49. Graves, A., M. G. Bellemare, J. Menick, R. Munos and K. Kavukcuoglu, "Automated Curriculum Learning for Neural Networks", *International Conference on Machine Learning*, pp. 1311–1320, Sydney, Australia, 2017.

50. Tsvetkov, Y., M. Faruqui, W. Ling, B. MacWhinney and C. Dyer, "Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning", *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1, pp. 130–139, Berlin, Germany, 2016.

51. Cirik, V., E. Hovy and L. Morency, "Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks", ArXiv:1611.06204, 2016.

52. Kumar, M., B. Packer and D. Koller, "Self-Paced Learning for Latent Variable Models", *Advances in Neural Information Processing Systems*, Vol. 23, 2010.

53. Soviany, P., R. T. Ionescu, P. Rota and N. Sebe, "Curriculum Learning: A Survey", *International Journal of Computer Vision*, Vol. 130, No. 6, pp. 1526–1565, 2022.

54. Wiering, M. A. and M. Van Otterlo, "Reinforcement Learning", *Adaptation, Learning, and Optimization*, Vol. 12, No. 3, p. 729, 2012.

55. Mathur, A. P. and N. O. Tippenhauer, "SWaT: A Water Treatment Testbed for Research and Training on ICS Security", *International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pp. 31–36, Vienna, Austria, 2016.

56. Goh, J., S. Adepu, K. N. Junejo and A. Mathur, "A Dataset to Support Research in the Design of Secure Water Treatment Systems", *11th International Critical Information Infrastructures Security Conference*, pp. 88–99, Paris, France, 2017.

57. Nagatsuka, K., C. Broni-Bediako and M. Atsumi, "Length-Based Curriculum Learning for Efficient Pre-training of Language Models", *New Generation Computing*, Vol. 41, No. 1, pp. 109–134, 2023.

58. Sun, C., A. Shrivastava, S. Singh and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 843–852, Venice, Italy, 2017.

59. Brigato, L. and L. Iocchi, "A Close Look at Deep Learning with Small Data", *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2490–2497, Milan, Italy, 2021.

60. Dickey, D. A. and W. A. Fuller, "Distribution of the Estimators for Autoregressive

Time Series with a Unit Root", *Journal of the American Statistical Association*, Vol. 74, No. 366a, pp. 427–431, 1979.

61. Paparoditis, E. and D. N. Politis, "The Asymptotic Size and Power of the Augmented Dickey–Fuller Test for a Unit Root", *Econometric Reviews*, Vol. 37, No. 9, pp. 955–973, 2018.

62. Schafer, R. W., "What is a Savitzky-Golay Filter? [Lecture Notes]", *IEEE Signal Processing Magazine*, Vol. 28, No. 4, pp. 111–117, 2011.

63. Hu, X., L. Chu, J. Pei, W. Liu and J. Bian, "Model Complexity of Deep Learning: A Survey", *Knowledge and Information Systems*, Vol. 63, pp. 2585–2619, 2021.

64. Bahdanau, D., K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", ArXiv:1409.0473, 2016.

65. Chen, Z., C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based Network Anomaly Detection", *2018 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, Phoenix, United States, 2018.

66. Zong, B., Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho and H. Chen, "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection", *International Conference on Learning Representations*, Vancouver, Canada, 2018.

67. Park, D., Y. Hoshi and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-assisted Feeding using an LSTM-based Variational Autoencoder", *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1544–1551, 2018.

68. Zhang, J. E., D. Wu and B. Boulet, "Time Series Anomaly Detection via Reinforcement Learning-Based Model Selection", *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 193–199, Halifax, Canada, 2022.

69. Duan, T., *Unsupervised Multivariate Time Series Anomaly Detection via Transformer-based Models and Time Series Encoding*, M.S Thesis, University of Toronto, 2021.

70. Kumar, L. A. and M. Senthilkumar, *Automation in Textile Machinery: Instrumentation and Control System Design Principles*, CRC Press, Florida, 2018.