End-to-End Anomaly Detection in Stream Data

by

Zahra Zohrevand

M.Sc., Sharif University of Technology
B.Sc., Bu-Ali Sina University

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Zahra Zohrevand**
Degree:	**Doctor of Philosophy**
Thesis title:	**End-to-End Anomaly Detection in Stream Data**
Committee:	**Chair:** Diana Cukierman University Lecturer, Computing Science
	Uwe Glässer Supervisor Professor, Computing Science
	Fred Popowich Committee Member Professor, Computing Science
	Martin Ester Examiner Professor, Computing Science
	David Skillicorn External Examiner Professor School of Computing Queen's University

Abstract

Nowadays, huge volumes of data are generated with increasing velocity through various systems, applications, and activities. This increases the demand for stream and time series analysis to react to changing conditions in real-time for enhanced efficiency and quality of service delivery as well as upgraded safety and security in private and public sectors. Despite its very rich history, time series anomaly detection is still one of the vital topics in machine learning research and is receiving increasing attention. Identifying hidden patterns and selecting an appropriate model that fits the observed data well and also carries over to unobserved data is not a trivial task. Due to the increasing diversity of data sources and associated stochastic processes, this pivotal data analysis topic is loaded with various challenges like complex latent patterns, concept drift, and overfitting that may mislead the model and cause a high false alarm rate. Handling these challenges leads the advanced anomaly detection methods to develop sophisticated decision logic, which turns them into mysterious and inexplicable black-boxes. Contrary to this trend, end-users expect transparency and verifiability to trust a model and the outcomes it produces. Also, pointing the users to the most anomalous/malicious areas of time series and causal features could save them time, energy, and money.

For the mentioned reasons, this thesis is addressing the crucial challenges in an end-to-end pipeline of stream-based anomaly detection through the three essential phases of behavior prediction, inference, and interpretation. The first step is focused on devising a time series model that leads to high average accuracy as well as small error deviation. On this basis, we propose higher-quality anomaly detection and scoring techniques that utilize the related contexts to reclassify the observations and post-pruning the unjustified events. Last but not least, we make the predictive process transparent and verifiable by providing meaningful reasoning behind its generated results based on the understandable concepts by a human. The provided insight can pinpoint the anomalous regions of time series and explain why the current status of a system has been flagged as anomalous.

Stream-based anomaly detection research is a principal area of innovation to support our economy, security, and even the safety and health of societies worldwide. We believe our proposed analysis techniques can contribute to building a situational awareness platform and open new perspectives in a variety of domains like cybersecurity, and health.

Keywords: time series; anomaly detection; time series anomaly detection; stream data; explainable AI; cybersecurity; deep learning

In loving memory of my father

To my mother, for her everlasting love and support

To my love, Mehdi

"The more I read, the more I acquire, the more certain I am that I know nothing."

—Voltaire

Acknowledgements

First and foremost, I offer my sincerest gratitude to my senior supervisor, Dr. Uwe Glässer; though, words are powerless to express my appreciation for his inspiring enthusiasm, patience, and continuous support throughout my PhD life with his supervision, understanding, and resourcefulness. I am wholeheartedly grateful for all his invaluable contributions of time and insight to make my research productive, stimulating, and exciting. Not only did I learn a lot of things in my research from him, but he also taught me invaluable life lessons.

I would like to sincerely thank Dr. Fred Popowich for his precious advice, guidance, and efforts on enriching various technical and practical aspects of this research. I would also like to thank Dr. David Skillikorn and Dr. Martin Ester for their thorough examination of this thesis and their suggestions, corrections, and remarks on this thesis.

I am forever indebted to all my co-authors, Dr. Uwe Glässer, Dr. Hamed Yaghoubi Shahir, Dr. Mohammad Ali Tayebi, Mehdi Shirmaleki, Ali Arab, Mehrdad Mansouri, Mahsa Keramati, Amir Yaghoubi Shahir, Saeed Arasteh, and Tilemachos Charalampous.

My friends at Simon Fraser University made my studies more pleasant and enjoyable. Especially, I am grateful to Maryam, Sima, Nazanin, Mahsa, Shadab, Sedigheh, Golnaz, Hossein, Hamed, Payam, Hamid, and Kiarash.

Heartfelt thanks goes to my parents and my wonderful brothers. I appreciate their endless love and support that made all these years a pleasant chapter of my life. My beloved family are always my sense of safety and calmness. A special thanks goes to my mother, a constant source of support and encouragement. Mum, you made an untold number of sacrifices for me; thank you for being so loving, caring and supportive.

Last but not least, I am immensely grateful to my spouse, Mehdi, for all his love, patience and kindness. His determination, confidence, and courage was inspiring for me and I could not have completed this work without his unfaltering love and support.

I would like to acknowledge the administrative and technical staff in the School of Computing Science for making this school a productive environment. I also wish to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and MDA Systems Ltd. for their financial support in the course of this research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Life is not starting today and is not finishing right away. We know of history and take advantage of the collected information over time in our current decisions and future resolutions. The same approach can be fulfilled in the data analysis context, especially in the big data era, which provides easy access to massive volumes of data from demographics, social networks, heterogeneous sensor networks, climate data, and so on.

Recent advances in technology have brought major breakthroughs in data collection, enabling a large amount of data to be gathered over time. Sets of observations that have been recorded in an orderly fashion and are correlated in time are called time series (TS) [30]. Time series analysis (TSA) comprises methods for analyzing time series data to extract meaningful knowledge and data characteristics that can be useful to track the targeted system's behavior and its evolution over time [126]. Keeping the end-users involved in this continuous analysis and monitoring process is time-consuming, expensive, and almost impossible. However, being aware of the changes and the events of interest is pivotal; this mission can be fulfilled utilizing anomaly detection (AD) in time series data, which examines anomalous behaviors across time [122]. Time series anomaly detection is essential and attracting significant interest in various event-sensitive scenarios such as causal disease discovery, robotic system monitoring, heterogeneous networks, and data center security [47,152,300]. For example, in the modern world of digital telecommunications, not only our communication and information are distributed through computer networks, but also electric power grids, transportation systems, public water utilities, like other critical infrastructure, routinely rely on automated control through network connections for their continuous operation. However, they are vulnerable to attack and are increasingly targeted with the aim to expose, alter, or steal information. Autonomous monitoring and tracking of anomalous behaviors is thus crucial for situational awareness and detecting suspicious events [10, 152, 292, 300].

Temporal anomaly detection is one of the primary approaches that can contribute to find and distinguish anomalies in the trace of observations. While general AD problem has a rich history within the data mining and statistical machine learning communities [35, 205], and a variety of stand-alone as well as ensemble methods have been proposed to handle this problem [55, 76], it is still one of the pivotal topics in the context of TS analysis [287]. A unique quality of temporal data

differentiating it from other data studied in the classical AD literature is the presence of dependencies among measurements induced by the temporal dimensions. Further, the observed system behavior is usually stochastic making modeling and anomaly detection a very tricky task, and thus often requiring customized methods.

1.1 Motivations and Research Objectives

As aforesaid the nature of large-scale systems, like critical infrastructure, demands situational awareness and understanding of the decision-maker. Therefore, as illustrated in Figure 1.1, this study mainly focuses on providing awareness through an end-to-end framework of anomaly/event detection in stream data. Also, most practical and real-world applications of stream data lack supervised information about the anomalies (e.g. how many are, where they are). Thus, approaches that require data labels, such as supervised classification-based methods, are naturally inappropriate for continuous learning and real-time anomaly detection [8]. Therefore, herein, we assume that the training data (or a majority of that) is normal and the AD model is supposed to discover future anomalies. The following main steps are required to realize our targeted situational awareness framework. In this process, we assume the system behavior can be represented using a multivariate time series X_T.

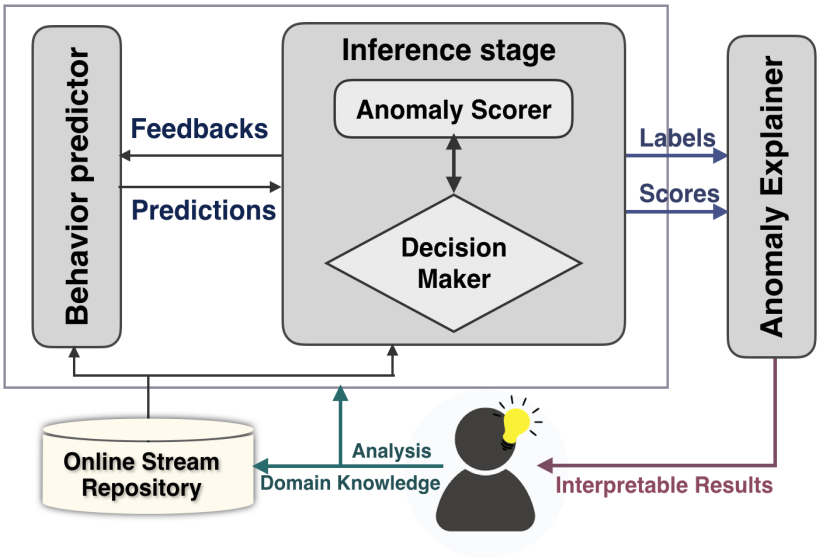


Figure 1.1: Schema of an end-to-end anomaly detection framework

Online behavior predictor. In the first place, we need to learn the knowns and normal behavior. Applying a predictor model [98, 196] capable of capturing the latent complex patterns within the data will unquestionably contribute to higher quality anomaly detection methods. However, identifying hidden patterns and selecting an appropriate model that fits the historical observations well and also carries over to unobserved data is not a trivial task. Also, in critical applications, a model is required to process millions, or even billions of time series data in near real time. So, efficiency is one of the major prerequisites for online AD models. Traditional statistical models [233] can easily be adopted

online, but their accuracies are not sufficient for industrial applications. Even though the models with large time complexity are good at accuracy, they are often impractical in online stream data analysis. Thus, devising an efficient TS model which leads to high average accuracy and small error deviation is one of the key objectives of this research.

$\Rightarrow$ An online behavior predictor is required to predict the next observations of a specific stochastic time series:

$$\text{TSF}(x_{t-w,\ldots,t}) = X_{T^+}$$

Online anomaly detection. AD is beyond modelling normal behavior. Increasingly diverse data sources and associated stochastic processes make this vitally important research topic more complex than ever. Further, the restrictive closed-world assumption [284] and the fundamental issues AD faces (e.g., clustered anomalies, masking, swamping) can mislead the model, generating a high false alarm rate [260]. Conversely, due to base-rate fallacy [221, 260], a small false-alarm rate in the train set can lead to significant false-alarm rates during monitoring of real systems— occasionally large enough to make the model futile. And surprisingly, a very small false-alarm rate means an overwhelming number of false-positive cases, especially for complex and critical systems. This is one of the notorious challenges that even state-of-the-art AI-based anomaly detection systems have not overcome, which may lead to serious short-term and long-term consequences. Because of the factors described, I target the false-alarm mitigation problem in an end-to-end framework and propose a higher-quality anomaly scoring technique by utilizing the related contexts to reclassify observations and post-prune unjustified events.

$\Rightarrow$ An online anomaly detection framework is needed to spot suspicious anomalous event:

$$ADF(x_{t+1}|X_T) \rightarrow [0, \infty)$$

Online anomaly explanation. An AD process should be transparent, ethical, trustable, and verifiable. To this end, the model needs to provide driving factors leading to the generated results based on understandable concepts by humans. Providing insights which enhances plausibility will keep users involved and let them trust a model's prediction. This especially matters when users should take action but are unsure of the associated risk. For example, it is not easy to interrupt or shut down a critical control system just because of some unknown and enigmatic risks based on vague threats predicted by normally accurate AI models. While sometimes it takes months or even years till the organizations determine the exact causes behind cyberattacks.

Most advanced AI methods, like deep neural networks, extract complex patterns without any explicit programming, turning them into intractable and uninterpretable black-boxes [78]. In other words, they imprison the knowledge and intuition and just provide us with the final results. In light of this, I formulate the explainability aspect of time series anomaly detection and also propose

temporal explainers to provide easy-to-understand and precise explanations of anomalous behavior in multi-dimensional, real-valued datasets concerning sequential relations between stream data.

$\Rightarrow$ An online explainer is needed to identify the most significant, conspicuous and ideally intuitive causal characteristics (F_i^*) associated with an arbitrary anomaly detector:

$$TADExp(W_i, \mathcal{BB}_{\mathcal{AD}}) \rightarrow {F_i}^*$$

1.2 Applications

The main focus of this thesis is novel/anomaly detection in stream data. However, *anomalies of interest* may, and often do, vary from one context to another. For example, differentiating anomalies of interest in a cybersecurity context, when dealing with *zero-day exploits* or other suspicious anomalous behavior indicating a potential security threat, from the set of those anomalies considered irrelevant to security turns out to be an even more intricate task than finding all the anomalies [260].

1.2.1 Cybersecurity application

To produce concrete results and evaluate the proposed algorithms' performance on complex systems, I consider tracing cyberattacks on critical infrastructure as the main application, specifically supervisory control and cyber-physical systems (CPS) used for the continuous operation of such infrastructure. Cyberattacks are becoming increasingly routine. Information security breaches frequently compromise protection of sensitive data and information, exposing personal identities [166, 247], intellectual property, or financial assets and potentially cause substantial damage that can ruin lives and businesses [1, 166]. Even more troublesome, such attacks go often unnoticed for extended periods of time. Still, things can get a lot worse when cyberattacks target supervisory control systems and CPSs essential for operating critical infrastructure on which we all depend in our daily lives, such as water management systems, power grids, and transportation systems. Besides temporal disruption of critical services, sophisticated attacks may cause physical damage of vital system components, and ultimately even threaten human safety. Despite the similarities, CPSs come with their own unique threat spaces because of the discrepancies in their basis and application [28].

Since CPSs and simple embedded controllers for operating critical infrastructure interact with processes in their physical environment by reading data signals and generating control signals in response, there are plenty of latent stochastic, or semi-stochastic, external factors or behavior instabilities that can influence the control process and complicate identifying hidden patterns [301]. Also, anomalies can originate from various sources like cyber or physical attacks, errors in control or communication networks, and other malfunctions with system-wide or local effects. Since fully immunizing these systems against cyberattacks is not feasible, it is crucial for organizations operating in public or private sectors to invigorate the detective, predictive, and preventive mechanisms to

mitigate the risk of disruptions, resource damage or loss. I therefore propose a methodical approach to cyber situation analysis for critical infrastructure by monitoring its behavior and raising a red-flag in the presence of any suspicious activity.

Cyber Situation Analysis Framework (CSAF). This study proposes a high-level architecture for CSAF based on two existing frameworks: *1)* STDF [167] which mainly focuses on information fusion and situation analysis processes; and *2)* CSOC [217] which identifies generic yet crucial components for cyber situational awareness. The main aspects of each of these two frameworks are outlined, followed by the proposed architecture.

- STDF (State Transition Data Fusion) framework—an extension of the JDL data fusion model [239]—provides a unification of both sensor and higher-level fusion across three principal levels [167]: object assessment, situation assessment, and impact assessment. Like the JDL model, STDF is a functional model, although less abstract than the JDL model, and seeks to demonstrate a unifying framework across the three levels.

- The CSOC (Cybersecurity Operations Centre) [217] framework is proposed as a model for security centers that comprise people who monitor ICT systems, infrastructure, applications and services. In [217], the main responsibilities of the CSOC are split into three major categories: collection, analysis and response. Generally speaking, *security operation centre* (e.g., CSOC and MSOC—maritime security operation centre) is a generic term describing the essential building blocks of a platform providing prediction, detection, and reaction services for security purposes [29]. Crucial elements of a CSOC framework include, but are not limited to: *1)* data collection components along with a processed data repository; *2)* incident analysis and knowledge base components; and, *3)* reaction and reporting components [29, 217].

Like STDF, our framework compromises three logically separated levels: *object assessment*, *situation assessment*, and *impact assessment*, where objects refer to individual evidence (e.g., logs from different sources), situation refers to behavior of the target system in the real-world, and impact is defined in terms of the severity level of threats or attacks. As illustrated in Figure 1.2, each level consists of a number of components. The first level is responsible for data collection from various sources. The second level is the analytic heart of CSAF and responsible for modeling the normal behavior of the target system to provide anomaly detection and behavior prediction services. The third and last level is responsible for assessing the impact of the detected/predicted situation as well as reporting the incidents to domain experts.

1.2.2 Generic applications

Even though, I exemplify and experimentally validate the proposed framework for the cybersecurity domain, the proposed algorithms can conceptually be applied to various real-world domains. Discovering unexpected events or rare patterns in sequential data is very important and popular in many other applications. For example, accurate online anomaly detection in industrial applications

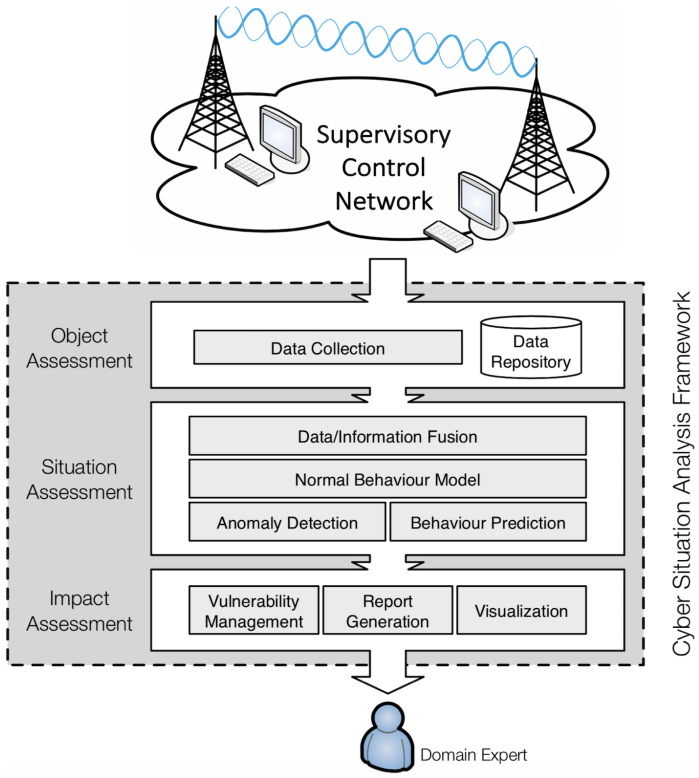


Figure 1.2: Cyber Situation Analysis Framework

can trigger troubleshooting, help to avoid loss in revenue, and maintain the reputation and branding for a company [233]. Some of the well-known applications of temporal anomaly detection can be described as follows [30, 51]:

- **Fraud detection.** It targets detecting criminal activities occurring in commercial organizations. The malicious activities may be caused by the actual or disguised customers of the organization. Fraud is any type of unauthorized access or consumption of the organization's resources. The organizations are interested in the immediate detection of such frauds to prevent economic losses. Some of the specific applications of fraud detection are in credit cards, mobile phones, insurance claims, and insider trading businesses.

- **Medical and public health anomaly detection.** The health domain is strongly entangled with patient records. Anomalies in this context can be caused due to several reasons, such as disease outbreak, abnormal patient conditions, instrumentation errors, or recording errors. This is an extremely critical and challenging domain with a very high cost of misclassification.

- **Industrial damage detection.** Most of the industrial units, including CPSs, suffer various damages due to continuous usage and typical corrosion. Early detection of such damages based on sensor data helps to prevent cascading damages or losses and also enhances the

system health management. Fault detection in mechanical units and structural defect detection are two major specific applications in this domain.

- **Anomaly detection in text data.** Novel topics, events, or new trends in social media, news articles, or other collections of documents is the main application of sequential anomaly detection in textual data. High-dimensionality, sparsity, and variety of documents are some of the challenges in this context.

- **Other domains.** There are a variety of other domains such as vision, speech recognition, robotic, traffic monitoring, click-through protection, web applications, biological data, census data, computational criminology, astronomical data, which take advantage of anomaly detection in their applications.

1.3 Thesis Contributions[1]

This research proposes novel algorithms in an end-to-end pipeline of temporal anomaly detection to address several crucial issues.

1. **An interpretable anomaly detection algorithm.** Interpretable and online anomaly detection algorithms are required to trace suspicious activities in critical infrastructures. Hidden Markovian models (HMM) are one of the very well-known family of techniques in the AD domain, but they suffer from several drawbacks. For example, applying log-likelihood to find anomalies is not robust to detect all the adversarial attacks. Also, existing HMM-based AD techniques do not specify the exact source of the anomaly. In Chapter 3, I address these problems and propose a novel hierarchical hidden semi-Markovian model (HSMM) framework. This research is published in [300].
   Most significant contributions: 1) A novel hierarchical model to follow the system behavior in different granularity levels, 2) A robust anomaly scoring technique utilizing the reference windows.

2. **A novel behavior modeling algorithm.** As mentioned, identifying hidden patterns and selecting an appropriate model that fits the observed data well and also carries over to unobserved data is not a trivial task. Devising a temporal forecasting model which leads to high average accuracy and small error deviation is one of the key objectives of this research. Beyond accuracy-related measures, this method is expected to produce high *quality* results, meaning a reliable estimation of the expectations and anomalies, in the presence of an unobserved/unknown event [35, 60, 96]. While current forecasting algorithms have achieved their highest levels of accuracy, the next level of improvement can be realized by combining several models

[1]Zahra Zohrevand has provided the main research and technical contributions of the referenced joint publications that she has been listed as their lead author.

with different properties. For example, the limitations of data-intensive techniques such as lacking a confidence interval or the possibility of generating off predictions can be covered by model-driven techniques. In Chapter 4, I propose a new hybrid method capable of handling complicated TS comprising linear and non-linear components. These results are published in [301]

Most significant contributions: 1) A novel hybrid time series forecasting technique to limit the error interval, 2) A novel multi-branch data augmentation technique robust to noise.

3. **Dynamic attack scoring using distributed local detectors.** Not all the novel or isolated events are *anomalies of interest*. This matter is domain-specific; for example, in the context of cybersecurity, differentiating suspicious anomalous behavior indicating a potential security threat from the set of those anomalies considered irrelevant to security is an even more intricate task than finding anomalies. There is a need to devise powerful anomaly scoring algorithms that maintain an acceptable trade-off between recall and false-alarm rates still exists [299]. To address this issue, in Chapter 5, I formulate the inference engine concept and propose a hierarchical platform to perform dynamic anomaly/attack scoring in distributed systems. This research is published in [298, 299].

   Most significant contributions: 1) Devised a method to ignore contextual noise and reduce false-alarm rates, 2) Orchestrated a network of detectors to detect collective attacks.

4. **Anomaly detection explanation.** Having an effective anomaly detection method is of great significance, and the interpretability of the underlying decision-making process matters. Causality inference is an implicit but entangled part of this task to address questions like why a given point or collection of points in a stream data is considered anomalous compared to other data points. This is while the increasing diversity of data sources, the complexity of hidden factors, and the continuous evolution of normal behavior push advanced anomaly detection techniques to develop sophisticated decision logic, which turns them into mysterious and unexplainable black-boxes. This leads to the prime focus of Chapter 6 which is explaining why a time series is flagged as anomalous. I propose a few techniques including model-agnostic and model-specific explainers to provide the required interpretation for both convincing the human analyst about the isolating factors and highlighting the potential model limitations. One of these studies is published in [197] and the rest will be submitted.

   Most significant contributions: 1) Formulated temporal anomaly detection explanation problem, 2) Proposed a novel explanation technique, 3) Overcame the fidelity-interpretability problem in finding the proximity measure of the explanation.

5. **Spatiotemporal situational awareness.** Spatiotemporal data differs from the previously analyzed time series because of having both space and time as the ubiquitous aspects of observations. This introduces another type of sequential dependencies among measurements induced by the spatial dimension, which have their own additional challenges that need to be dealt

with. In Chapter 7, we present two projects focusing on spatiotemporal aspect in the context of maritime domain awareness. These studies are published in [14, 253].

Thesis structure. The remainder of this thesis is organized as follows. Chapter 2 explains the basic concepts, formulates and justifies a collection of requirements which should be addressed in anomaly detection. Subsequently, Chapters 3, 4, 5, and 6 explain the four main phases of this study. For each phase, I first set out the problem definition and then provide an informal description of the background and the challenges targeted by the proposed method in that Section. Then, Chapter 7 explores the world of spatiotemporal data and briefly explains our proposed analytical algorithms toward maritime domain awareness. Finally, Chapter 8 briefly highlights the key lessons learned and outlines directions for future research.

Chapter 2

Background

In this chapter, I present the basic definitions of time series (TS) and then drill down into different meanings of anomalies. Following this, I will outline anomaly detection (AD) techniques and describe other key concepts and requirements involved in anomaly scoring and false alarm mitigation for AD systems.

2.1 Time Series

It is important to have a formal definition of time series (TS) to present the properties and algorithms based on the same notation. TS are interpreted as consecutive observations at relative time points $X = \langle x_t : t \in T \rangle$, $T \subseteq \mathbb{N}$. Formally, each x_t is an m-dimensional vector, $x_t \in \mathbb{R}^m$, where x_t^k refers to the value of the kth variable of x_t for $1 \leq k \leq m$. Mathematically speaking, TS can be categorized as linear, or non-linear (see Figure 2.1). A model for a TS (O_t) is *linear* if it can be expressed as $o_t = \alpha_0 + \alpha_1 u_{1,t} + \alpha_2 u_{2,t} + \ldots + \alpha_m u_{m,t} + z_t$, where $u_{i,t}$ is the value of the ith predictor (or explanatory) variable at time t; z_t is the error at time t; and $\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_m$ are model parameters, which can be estimated by least squares. *Non-linear* TS are those that cannot be modeled by a linear formula.

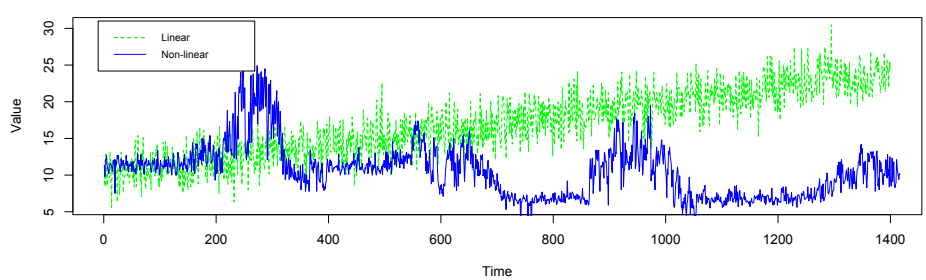


Figure 2.1: Example of linear and nonlinear TS

Randomness may also be used to categorize TS as *non-stochastic* (i.e., deterministic) or *stochastic* (i.e., non-deterministic). Non-stochastic TS are a collection of non-random variables that can be described by explicit mathematical relations. Thus, predicting exact future values is possible within a non-stochastic TS. Non-stochastic TS can be categorized by periodicity:

- *Periodic*: A TS is composed of repeated values in a regular period or intervals
- *Non-periodic*: There is no explicit pattern in the TS data. This property can happen in the following two ways of almost periodic and transient non-periodic.

Conversely, stochastic TS are a collection of random variables. Each variable is uniquely associated with an element, and is indexed by time. Forecasting the exact value of a sequence is impossible due to randomness. Stochastic TS can further be categorized by stationarity:

- *Stationary* sequences are those in which statistical properties such as mean, variance, and/or autocorrelation stay constant over time. Most statistical methods are applicable to (almost) stationary TS.
- *Non-stationary* sequences are those in which statistical properties change over time. Most temporal, real-world data are highly random and mainly non-stationary due to the influence of external factors such as human interaction.

As aforementioned, most of the temporal real-world data obtained from monitoring systems, social networks, financial data, and other datasets of interest, which are influenced by external factors and have interaction with the human being, are highly random and mainly non-stationary.

2.2 Anomaly Definition

A general definition of anomaly (outlier) builds on Hawkins' statement in [128]: "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". This abstract view is ambiguous and it can lead to different AD systems, based on the following interpretations:

1. **Rare events**, because anomalies do not happen frequently.
2. **Distinct events**, because anomalies are odd.
3. **Abnormal events**, because anomalies diverge from normal expectations.

However, these definitions are not comprehensive enough to account for all anomalies or suspicious events. Neither rarity, distinctness, nor abnormality are meaningful without adopting a threshold; however, an ill-defined or otherwise poorly estimated threshold may result in a huge rate of erroneous readings (i.e., false positives/negatives). Further, methods which focus on rarity scores are often incomparable because rarity is highly dependent on the AD algorithm. Other methods such as clustering [224], which focus on distance to compute anomaly scores (i.e., the distance of an anomaly to the closest clusters, or the sparsity of the cluster to which it belongs), assume distance or density are meaningful when this may not be the case. Also, the normal data should be labelled, while it is not the case in many target domains like IDS [91].

An ideal definition should be applicable to all of the possible distributions without need for additional stipulations [91]. Moreover, it should enable comparison of anomality across variables. Accordingly, I propose a new definition for anomaly, within the context of AD: "An event is considered a suspicious anomaly if it is highly distinct in terms of feature values, or clustered but unobserved previously; and persistent or close to previous suspicious cases to reduce their noise likelihood."

Anomalies categories. Anomalies are classified into the following three categories:

- *Point anomalies,* which refers to a data point that is obviously distinguishable from the other data points.
- *Contextual anomalies,* which refers to a data point that is anomalous only in a specific context and normal in all other contexts.
- *Collective anomalies,* which refers data points that may not be anomalous individually, but are anomalous when they occur collectively.

However, these categorizations are not mutually exclusive. For example, a contextual anomaly may refer to a point anomaly, or a collective anomaly. Further, from a statistical point of view, the TS anomaly detection problem is formulated as finding data points which are different relative to some standard or usual signal. Traditionally, certain types of TS anomalies are more important, and include [4]:

- *Additive anomalies*, which refer to any unexpected growth within a short period, resembling a spike. These are comprised of point anomalies.
- *Temporal changes*, which refer to any short-term differences in target metric behaviours. These are comprised of collective anomalies, though they may comprise additive anomalies up to a point.
- *Level or seasonal shifts*, which refer to changes in the total value of the metric over a long period. These usually do not change the shape of the target signal.

2.3 Anomaly Detection

As the name implies, AD is concerned with detecting anomalies (i.e., outliers) within a data set. AD is also referred to as outlier detection, event detection, novelty detection, deviant discovery, changepoint detection, fault detection, intrusion detection, and misuse detection. While the aim of anomaly detection is straightforward, the definition is not transparent. Moreover, AD can be challenging because of various reasons such as lack of knowledge, ambiguous normal region (see Table 2.1 and Section 2.4). Further, there are unique challenges associated with AD of temporal data (see Table 2.1). Multiple methods have been developed to perform AD, including classification, clustering, nearest

neighbour, density, statistical methods, information theory, spectral decomposition, visualization, and signal processing. These techniques can be classified as:

- *Supervised methods*, in which all train labels are accessible. To this end, we can apply any classification method to handle imbalanced datasets like Support Vector Machines (SVM), or Artificial Neural Networks (ANN). These methods are inefficient for real life because real-world data is dynamic and often we are not aware of anomaly cases.

- *Semi-supervised methods*, in which a majority of the data is labelled, while a minority is not. One popular model, one-class classification, presumes there are no anomalies in the training set. The model is trained based on all normal cases and anomalies are distinguished based on their deviation from the trained model. Other, popular semi-supervised methods include autoencoders, one-class SVM, and probability density estimators, such as kernel density estimation or Gaussian mixture-models.

- *Unsupervised methods*, in which no data label is available. This family of methods assign anomaly scores to data points based on the configuration of the data's features. These include neighbourhood-based methods like k-nearest neighbours algorithm; clustering-based methods like cluster-based local outlier factor; statistical-based like histogram-based outlier detection; and subspace-based methods like robust principal component analysis.

Anomaly scoring. An anomaly detection system (ADS) aims to order anomalies based on their anomalous score. One basic method transfers the original order in feature space through a scoring function $S_{AD} : X \rightarrow R+$, which assigns smaller scores to more anomalous points [302].

2.4 Challenges in Anomaly Detection

As mentioned earlier, anomaly detection and scoring are challenging for many reasons including high dimensional spaces, stochastic behavior, potential data drift, seasonality or highly irregular data observation rates, uncertain environments, mixed data types (e.g., continuous, integer, or lattice data), limited available data, varying lags in the emergence of anomalous behavior in one dimension compared to the others, and unknown hidden factors. Therefore, strong anomaly detection and scoring methods should be sufficiently capable and flexible to address these challenges using observed data [275]. AD methods are mainly evaluated according to their *detection rate* (i.e., the ratio of correctly detected anomalies to the total), and *false-alarm rate* (i.e., the ratio of misclassified normal data points to the total number of normal points). However, by focusing on meta-learning for AD algorithms [81, 82], I have devised and summarized criteria for anomaly scoring as well as corresponding measures to evaluate the strength and performance of these algorithms in addressing the specific circumstances of cyberattack detection and protection.

Challenges	**Anomaly detection**	**TS anomaly detection**
Open-ended definition	✓	✓
Being interpretable and explainable	✓	✓
Ambiguous definition of the normal regions (No precise boundary between normal and anomalies)	✓	✓
Robust to noises	✓	✓
Scale and dynamics	✓	✓
Challenging and ambiguous validation	✓	✓
Complexity and dynamicity of data	✓	✓
Lack of knowledge and noisy labels	✓	✓
Imbalance nature of data (rare vs common)	✓	✓
Asymmetric error (Application-based trade off between precision and recall)	✓	✓
Considering time dependencies	○	✓
Handle data evolution (including trend drifts)	○	✓
Being time efficient in stream data analysis	○	✓
Managing distributed data streams	○	✓
Handling complex and nonlinear patterns	○	✓
Deriving confidence interval	○	✓
Handling massive scale of history	○	✓

Table 2.1: Anomaly detection challenges

2.4.1 Masking effect

One anomalous point *masks* a second anomaly if the latter can be considered an anomaly only by itself but not in the presence of the first point. Thus, a masking effect may occur if the estimated mean and covariance are skewed toward a cluster of outlying observations such that the outlying point is not sufficiently far from the mean so that it can be detected [25].

A toy example of this scenario, which involves a method that dynamically updates its threshold value, is shown in Figure 2.2. Here, an attacker, who has probably realized the strategy of this AD algorithm, takes a gradual data poisoning strategy; in such an approach, the attacker intentionally feeds the system fake data that are similar enough to the normal process to be included in the confidence interval but slightly different to make the system shift its threshold value.

Thus, the attacker will subsequently be able to perform a masked attack (A) without being captured by the manipulated AD algorithm. In another example of masking in Figure 2.3, a rarity-based AD algorithm has missed A and B because of the limited number of anomalous points. In other words, if the method only looks for a very small rate of isolated cases, clustered anomalous points may influence the statistics so that none of them be declared as anomalies [184]. One statistical representation of this phenomenon is when $\sigma_A^2 \leq \sigma_N^2$, where σ_A^2 and σ_N^2 are the normalized sample variance of the selected anomaly points and selected normal points, respectively.

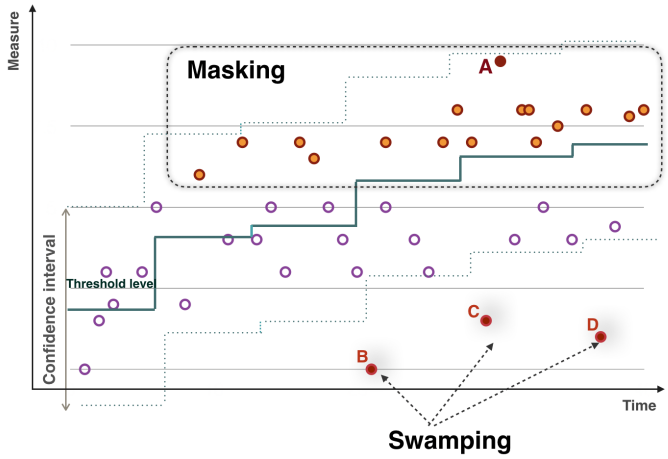


Figure 2.2: A simple example of Masking and Swamping effects

2.4.2 Swamping effect

The *swamping effect* is the reverse of the masking effect and occurs if the swamped data point can be considered an anomaly only in the presence of another data point(s). The false-positive cases *B, A,* and *D*, illustrated in Figure 2.2, which have been swamped by the orange observations are some examples of this phenomenon. If a group of outlying instances skews the mean and covariance estimates toward itself, it can swamp some non-outlying instances by increasing their distance from the shifted mean such that they are isolated as anomalies [25]. If a statistical test or AD algorithm

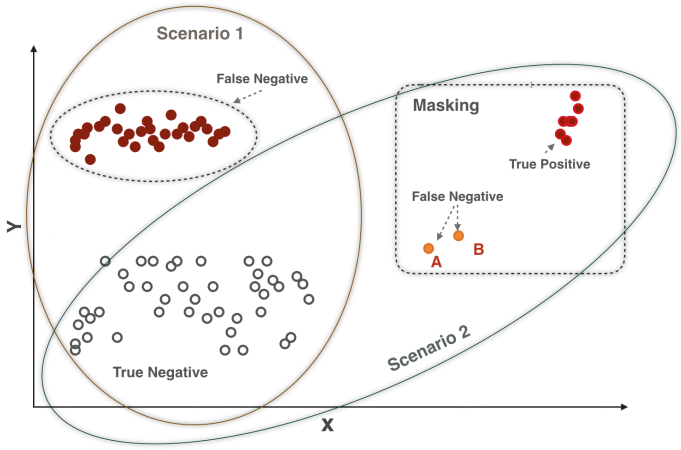


Figure 2.3: An example of rarity assumption based masking

overestimates the number of anomalies in a dataset, it might be influenced by the swamping effect. Therefore, any type of AD algorithm must be fine-tuned to avoid falling in such traps.

2.4.3 Variable frequency of anomalies

If anomalies are highly rare (1–10% of the whole observations), the rarity based techniques can perform very well. However, in some difficult configurations, more than 30% of data might be anomalous [154, 184], like DoS attacks, which happen more frequently than some other specific individual normal scenarios. For example, in *Scenario 1* shown in Figure 2.3, the AD algorithm has missed a group of anomalies only due to the rarity assumption. In such circumstances, the rarity assumption is a failed theory, so the group of methods that learn the boundary of normal behavior to highlight the anomalous cases might perform better.

Hence, the **relative frequency** is equal to the **contamination rate** or **plurality** as the proportion of anomalous incoming data points [82]. The reliability of an algorithm under variable frequency conditions is measurable based on its tolerance level under different levels of relative frequency without losing accuracy.

2.4.4 Curse of dimensionality

Having access to more features and detectors decreases the risk of losing influential information when performing the task of interest, but may cause other problems as follows [297]:

- *Irrelevant features.* A significant number of attributes may be irrelevant.
- *Concentration of scores and distances.* Numerical measures such as distance might be extremely similar.

- *Incomparable and uninterpretable scores.* Obtained scores produced in each domain are incomparable with the others, resulting in a final score that is neither strong nor semantically meaningful.
- *Exponential search space.* A non-systematically analyzable search space leads to an enormous possible hypothesis for every observed significance.

Although there are existing methods that can resolve one or more of these problems, some challenges remain as open research questions. From a statistical perspective, each irrelevant feature increases the dimensionality of the space. This phenomenon causes a problem in (naïve) density estimation methods that their required sample size scales exponentially with dimensionality, which is not always available. Moreover, having irrelevant features decreases the recall and precision. On the one hand, as the dimensionality of the data increases, anomalous data points may be covered under the similarity of the other unrelated and unimportant dimensions. On the other hand, the data domain space grows with dimensionality which may cause normal cases to fall apart from the others and be labeled anomalous. In other words, from a statistical perspective, the possibility of having more tails increases, which might consequently push the normal points towards the tails of some unimportant feature distributions [81].

2.4.5 Lag of emergence

Different parts of a complex system might react to an anomalous input very differently. For instance, some of the features might react to the occurred event sooner than the others. Consequently, the AD algorithm might bombard the end-user with the same anomaly on the time span, which is not desirable.

This phenomenon might happen because of various reasons say feature correlations and interactions, potential influential hidden factors, temporal relations. For instance, the response time of a computational system is highly correlated to a memory leak, through causality relation based on process footprints. So, a memory leakage event is first recognizable due to a spike in the process footprint; then, the increased response times caused by the frequent process swapping.

Also, having a very irregular rate of sampling in different dimensions is another potential reason for such lags. Thus, features with a smaller sampling rate (longer interval) might indicate the extreme or anomalous events, which have been already deciphered by analyzing other high-frequency features [275].

2.4.6 Domain specific criteria

Defining domain-specific measures may help evaluate ADS capabilities in very specific target domains. For example, [172] applies "*burst detection rate (BDR)*" to capture potential bursts which indicate attacks involving many network connections. This measure represents the ratio of the total number of intrusive network connections with a higher score than the threshold to the total number of intrusive network connections within attack intervals.

2.5 Anomaly Detection Approaches

The most important AD approaches are reviewed in the following sections. Each section corresponds to an AD category [2, 122, 264, 271], and each category groups AD techniques according to criteria. These criteria include, though are not limited to, the type and distribution of input data; the flexibility of the learning phase; the availability of data and resource constraints introduced by the application domain; and the availability of target labels.

2.5.1 Model-based and data-driven categorization

At the macro level, TS anomaly detection methods can be classified into *model-based* and *data-driven* strategies. Model-based categorization refers to a group of methods that use an existing framework to estimate data distribution. Data-driven categorization refers to methods that extract patterns from training data. Essentially, model-based category represents a top-down approach while data-driven represents a bottom-up approach. One issue with this categorization is model-based and data-driven groups are not mutually exclusive. Rather than positioning these techniques in a binary relationship, they are more accurately represented as a spectrum. Thus, traditional model-based (e.g., parametric statistical) methods are one end of the spectrum, while fully data-driven (e.g., deep learning) methods are at the other.

2.5.2 Technique-based categorization

Technique-based categorization is grounded on a combination of underlying techniques and measures. The main weakness of this categorization type is its lack of comprehensiveness; for example, several methods such as deep learning are not included in this categorization. Moreover, several methods can be included in more than one category. For example, there are many clustering-based AD methods that take advantage of the sliding window [140].

Statistical anomaly detection

These methods have some assumptions about normal observations, like data distribution restrictions. Therefore, anomalies are points with a low probability of generation by overall presumed distribution. Statistical anomaly detection methods can be divided into two major groups:

- Parametric methods, which lack of knowledge about unobserved data and the high-probability of behaviour-change over time, making them impractical to use for stream data.
- Non-parametric methods, in which the model learns the systems normal behaviour from the input data. There is greater potential to apply these methods to stream data, however not for high-dimensional scenarios.

Depth-based anomaly detection

This technique is a variant of statistical methods applicable to quantitative and low-dimensional data. Instead of considering statistical distribution, depth-based anomaly detection highlights anomalies based on the border of data space [242]. Each data point is represented in an n dimensional space with an assigned depth. According to the assigned depth, data points are organized into convex hull layers, and shallow depth values are declared anomalies.

Distance-based anomaly detection

In this approach, anomality is scored based on the distance of data points to k-nearest neighbors in a time window of length w. This approach is limited to data points in which distance computation is possible.

Deviation-based anomaly detection

This approach marks as anomalous those points which do not fit the general characteristics of the given set, thereby minimizing variance by removing anomalies. For example, a histogram-based method may be applied to decrease deviation [142]; however, this approach is not applicable to multivariate or distributed data.

Density-based anomaly detection

Density around a data point is compared with density around its local neighbours. Generally, any points far from dense areas are classified as anomalous. One method, Local Outlier Factors (LOF), computes the ratio of the local density of the target point versus its neighbors. A related method, Incremental LOF [123], has the same performance as standard LOF and can detect changes in data behaviour, making it more suitable for stream data. However, this method cannot distinguish new normal behaviour (resulting from potential trends or pattern shifts) from a real anomaly [44]. Generally speaking, density-based methods are impractical in presence of high dimensional datasets, because the accuracy of density estimation decreases with the increase of the number of dimensions. Moreover, while they are more successful than distance-based methods, they are also more expensive and computationally complicated.

Sliding window-based anomaly detection

The aim of this approach is to maintain statistical information on past observations. Choosing the best window-size is challenging, because a large window may cover outdated information and negatively decrease the importance of the recent points. Conversely, an overly small window-size may risk the loss of historical and statistical information like seasonality and cyclic patterns.

Clustering-based anomaly detection

Normal observations are usually clustered into several groups, while anomalous cases do not belong to any cluster. Generally, anomalies are far from their closest centroid; or they belong to small or sparse clusters while normal observations belong to large and dense clusters. This approach may encounter two main challenges: (1) Most clustering methods prioritize finding clusters rather than anomalies, and are thus not fully optimized for AD, (2) Most clustering algorithms require the number and shape of the target clusters, however the number of clusters is not definite in stream data.

2.5.3 Strategy-based categorization of anomaly detection techniques

Some studies argue that TS data are very similar to discrete sequential data, and thus shared methods can be applied in both contexts; however, this is not the case given basic differences between these types of data. TS include numeric data, while discrete sequences are symbols across time. Further, TS anomalies are usually detected based on their forecasts, while discrete data can be handled in various ways similar to distance anomalies. TS data are primarily explored using techniques like AR, VAR, ARIMA models, et cetera, while discrete sequences are explored using data mining techniques such as Markov models and deep learning. These datasets can be transformed into one another; for example, TS can be discretized to sequences to take advantage of sequence analysis methods. Moreover, techniques such as clustering can be used for both types of data.

AD methods appropriate for temporal data and their main approaches are described here. They are ordered based on the fundamental strategy and the type of data taken as input by methods [2]. The following can be grouped into two main families based on their input and goal: (1) Input: TS database, Goal: Identify anomalous sequences or subsequences; (2) Input: One TS, Goal: Identify anomalous point or subsequences.

Anomalies in TS databases

Several methods for AD in TS databases compute anomaly scores based on the entire sequence, while others do so by aggregating scores through overlapped fixed-size windows. These differences are explained further in the follows.

Direct detection of anomalous TS. These AD approaches are predicated on the assumption "anomalies are rare events." By training a model based on an entire database, it learns the normal behaviour and assigns a specific score to each TS, which can highlight anomalies. Multiple algorithms can be included in this category, some of which are included below.

Unsupervised discriminative approaches. The defining features of these approaches are similarity functions and the measures used to determine similarity. Discrimination is based on two principle rules: within-cluster similarity should be maximized, while between-cluster similarity should remain minimized. The anomaly score of a test sequence is defined as the distance to the centroid (or medoid) of the closest cluster. The primary variations across approaches in this group are due to differences of

the similarity measures and/or the clustering mechanism. The overall algorithm can be summarized as:

1. Compute clustering space (features)
2. Define a similarity function to compare two TS
3. Cluster the sequences using clustering algorithms based on the data type.
4. Obtain anomaly score of each test sequence

The anomaly score is defined based on the distance of each data point to the closest centroid or medoid. Two popular similarity measures include simple match count [169], and the normalized length of the longest common subsequence (LCS) [41]. The former measure is very efficient, while the latter is more computationally expensive; however, the latter is also applicable to segments in noisy sequences. Numerous studies have applied optimized versions of this technique for TS anomaly detection, including:

- nLCS with kmedoids [41, 42];
- nLCS with a variant of KNN [52], in which the anomaly score was computed based on the datapoint distance to the kth closest sequence;
- Count-based sequence similarity evaluation of the window-based subsequences [168];
- Window-based subsequences with K-Means [209];
- Phased K-Means [232] ;
- One-class SVM [88, 193] and one-class SVM with discretized data [268];
- Kernel-based feature maps of raw data with kNN or one-class SVM [86];
- Self-organizing maps with windowed subsequences;

Unsupervised parametric approaches. The overall process of methods in this category is based on computing the probability of any data point in the series based on the values at the previous few time steps. Therefore, the TS anomaly score is a function of its datapoint's anomaly score, so that a point with a very low generation probability will be marked as an anomaly. Three main high-level methods in this category are Finite State Automata (FSA), Markov Models, and Hidden Markov Models (HMM), which cover many anomaly detection studies and applications in various areas like intrusion detection and speech recognition.

The overall process of these methods is based on computing the probability of any data point in the series based on the values at the previous few time steps. The TS anomaly score is a function of its datapoints anomaly score, so a point with a very low generation probability is marked as an anomaly. Three main high-level methods in this category are Finite State Automata (FSA), Markov

Models, and Hidden Markov Models (HMM). These include many AD studies and applications in various areas such as intrusion detection and speech recognition [86, 281].

Finite state automata. These methods are applicable to fixed-size subsequences. After training FSA, all possible subsequences with the same length are extracted from the test TS and fed into FSA. If FSA ends in a state that does not have an outgoing edge to the next value in the test sequence, it is labeled as an anomaly (see figure 2.4) [52]. It is important to note that generating FSA is highly dependent on the application domain.

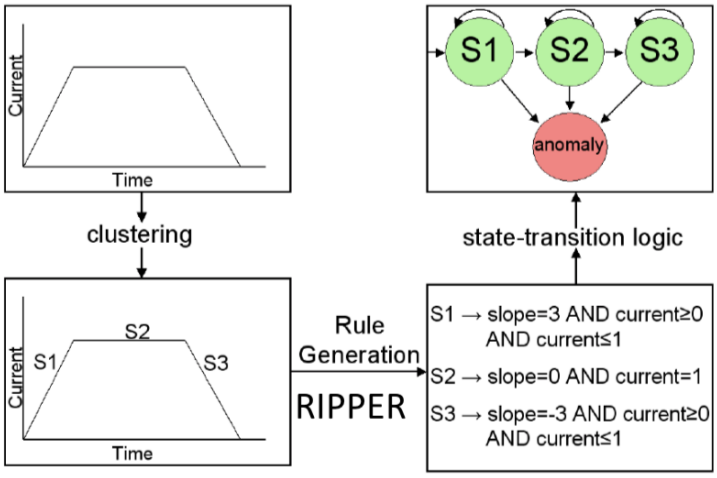


Figure 2.4: FSA-based anomaly detection (adapted from [52])

These approaches have been used in multiple studies [52, 201, 245, 288], and all follow these general steps:

- If the next observation matches the current states characteristics, remain in the current state;
- If the next observation matches the next states characteristics, transition to the next state;
- If the next observation matches neither of the current nor the next one, transition to an anomaly state.

Markov model-based anomaly detection. Markov models use generation probability to obtain the conditional probability of the observed symbols, their transition to each other and to detect anomaly series. There are different approaches to produce and preserve this distribution probability information; for example, probabilistic suffix trees (PSTs) are an efficient method [266] . The other advantage of Markov models is they do not limit the size of the historical subsequences, and thus the history size may be fixed, variable, or selective. Selective-size Markovian techniques, also be referred to as sparse Markovian techniques, compute conditional probability of the sequence through previous l symbols; however, they do not need to be continuous or last l preceding items of the current one [86].

Hidden Markov model-based anomaly detection. An HMM is a probabilistic model in which the system being modeled is represented as a Markov process with hidden states. Unlike the simple Markov model, in an HMM the state is not explicitly visible, while the output depends on the state. Even though stronger than the previous two methods, HMMs suffer a few drawbacks; they are not easily scalable to real-world datasets; their training usually needs significant manual intervention, experience with the data; and judicious selection of the model, the parameters, and initialization values of the parameters. Nevertheless, because they are highly interpretable and theoretically well-motivated, HMMs are one of the most widely used AD approaches and have been applied across various domains. Using HMM for intrusion detection was first proposed by Warrender et al. [281], in which they proposed an HMM-based approach to detect anomalous program traces in operating system calls data.

Supervised approaches. As previously mentioned, supervised approaches are those in which all train labels are accessible. Thus, various methods can be applied to the dataset, including: Positional system calls features with the RIPPER classifier [175]; Subsequences of positive and negative strings of behavior as features with string matching classifier [45, 111]; Bag of system calls features with decision tree, Naive Bayes, SVMs [145], Classifier neural networks [66]; Elman network [107].

Window-based detection of TS anomalies

Window-based approaches break sequences down into the same size overlapping subsequences, as illustrated in Figure 2.5. Thus, the anomaly score of each test sequence is computed based on the aggregation of anomaly scores of the underlying windows. There are advantages and disadvantages to these approaches. Unlike previously mentioned approaches, window-based methods can localize anomalies in smaller sequences, and thereby do not mark the entire sequence as an anomaly. However, this requires finding the best window length which adds another layer of complexity.

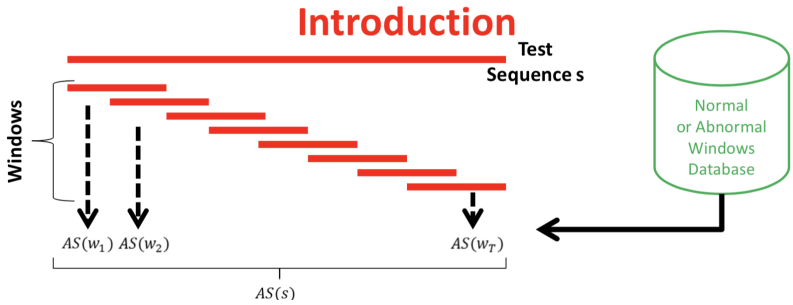


Figure 2.5: Window-based anomaly detection (adapted from [122])

Windows may also be referred to as look-back windows, fingerprints, pattern fragments, n-grams, sliding windows, motifs, et cetera. Window-based methods can be divided into the following:

1. Preceding based normal repository. The aim of these approaches are to extract and maintain a normal repository of subsequences, with size w, based on the training set. The repository is generated based on the preceding items. Anomaly scores are based on the degree to which each test sequence

matches existing subsequences in the repository. Sequence Time-Delay Embedding (STIDE) [133] is one such method that follows these general steps: (1) Divide the normal sequences into size k overlapping windows; (2) Obtain size k subsequences of the test sequence; (3) Subsequences that do not occur in the normal database are considered a mismatch. Now, if a test sequence has a large number of mismatches it is marked as an anomaly; however, if a test sequence is not in the database, the mismatch score is computed as the minimum Hamming distance between the window and any of the subsequences in the normal database normalized by k. This method, with slight variation, has been applied in several other ways as well [45, 83, 99, 106, 107].

2. Look-ahead based normal repository. Time-Delay Embedding (TIDE) [95] extracts patterns out of look-ahead items, and follow these general steps:

- Record the elements occurring at the distance 1, 2, ..., k in the sequence, for every element in every normal sequence;
- Create a normal database of such occurrences;
- Given a test sequence, find a look-ahead of the same size k;
- Compute the number of mismatches by checking each pair of element occurrence with the normal database;
- Score anomalies based on the number of mismatches normalized by the total number of such occurrence pairs.

3. Negative/mixed pattern database approaches. These methods create a repository of anomaly sequences and evaluate the test subsequences against this repository. They follow these approximate steps: (1) Extract all of the normal subsequences of length w from the input; (2) Find all subsequences with size w not in the set, and label them as detectors or negative cases [65, 66, 111]; (3) If a test sequence matches any detector in the repository it is considered an anomaly.

Detectors, which are included in the anomaly subsequences repository, should be far enough from normal cases and maximize the coverage of the non-self space. Different approaches can be applied to find these detectors which include extracting subsequences with at least r contiguous positions difference [65, 75]; having at least r different contiguous chunks [66]; or finding detectors in an n-dimensional hypersphere with $r \geq 1$ relative to all normal cases [111].

2.5.4 Anomalous subsequences in a test time series

This represents a collection of AD methods, with the aim of finding anomalous patterns or subsequences in a given test TS. Given a previously observed TS, the frequency of an anomalous subsequence in the test series will be substantially far from the expected value. Much research has applied different methods with this principle idea; for example, the TARZAN algorithm [151, 179, 180]:

- Builds a suffix tree based on discretized test and reference trees;

- Calculates the adjusted frequency of occurrence in both reference and test sequences for each subsequence in the test string;
- Checks the subsequence as anomalous if the difference of frequencies is greater than the threshold;
- Picks the longest set of subsequences from the reference that covers w; and uses a Markovian method to estimate its frequency in the test sequence, if w in the test sequence does not occur in the reference.

There are other methods with more relaxed matching criteria [17, 125] that evaluate and accept matching of any permutations of subsequence.

2.5.5 Anomalies within a given time series

Given a single TS, particular datapoints or subsequence as anomalies within the TS can be found. The primary difference between this category and the previous one is the input source (i.e., a singular rather than multiple TS) and consequently the algorithm. These methods identify anomalies in finer granularity as anomaly data points or subsequences rather than labelling the whole sequence as an anomaly. Consequently, these methods are more suitable for finding anomalies in stream data. Proposed methods in this group can be categorized as: (1) Point anomaly detection methods; (2) Subsequence anomaly detection methods.

Point anomaly detection methods

These include multiple approaches that look for anomalous points in TS.

Prediction based approaches compute the anomaly score of a point in the TS based on its deviance from the predicted values by the prediction model. There are various prediction models that have been applied, including median-based predictors [22]; average-based predictors; single-layer linear network predictors; multilayer perceptron (MLP) predictor [129]; deep learning predictors [98]; and support vector regression [192].

Profiling based approaches maintain a normal profile and compare the value of the upcoming time point against this profile to identify an anomaly. For example, the normal profile is tracked, based on the smoothed version of the historical TS and a variance vector [283]. An anomaly score is assigned after a new point is compared with the normal profile and the variance vector. Others [256] have used an ML model to maintain the normal profile, and new points are evaluated based on estimation of the next value.

Deviant based approaches find the points in a given TS, whose removal from the series decreases the error bound of representative histogram [142]. The number of buckets can be reduced to account for separate storage of these deviated points. To solve the AD problem, a dynamic programming mechanism with recursive relation, which breaks down the optimal set of k deviants to the deviants in the l highest or $k - l$ lowest values, has been proposed [142]. Another approach is an approximation

method, which finds a partial solution for a few scattered indices of the TS rather than for each value [207].

Subsequence anomaly detection methods

In some instances, detecting subsequences of anomalies is more important than focusing on specific points. This group of methods is highly practical for identifying collective anomaly patterns.

Discord discovery is a method in which the subsequence s of length l is called a discord (or anomaly) if it has the greatest distance to its nearest non-overlapping match [150]. The brute-force solution considers all possible subsequences and computes the distance of each one with every other non-overlapping subsequences of S. There are several methods to decrease the time complexity of this technique, such as applying top-k pruning; or heuristic methods for reordering candidates using Symbolic Aggregate Approximation (SAX) [149] and locality sensitive hashing [282].

Wavelet and Augmented Trie (WAT) [40] applies Haar Wavelet transform and symbol word mapping onto the TS and generates a prefix tree. Following this the algorithm applies a breadth-first search on the obtained tree to find an approximation of all candidate discord subsequences.

Multi-scale anomaly detection aims to find anomalies in several scales. Some can handle irregular TS analysis; for example, irregular cases are handled by defining a pattern as a subsequence of two consecutive points [282]. If the pattern is rare in respect to its slope and length relative to the other patterns, it is labelled an anomaly. Others [149] determine anomalous subsequences based on the level of their similarity with previous parts of the sequence. This method uses SAX to produce symbols and in the next step, a chaos bitmap is used for evaluating similarity.

2.6 Advanced Temporal Anomaly Detection Techniques

Since origination of Artificial Neural Networks (ANN) in 1943 and subsequent advancement of advanced deep learning, advanced temporal AD techniques have solved various problems in diverse contexts such as robotic processing; object recognition; speech recognition; handwriting classification; and even real-time sign-language translation [98]. Common and popular deep structures include the "Elman and Jordan Neural Network", "Convolutional Neural Network (CNN)", "Fully Convolutional Networks (FCN)", "Recurrent Neural Networks (RNN)", and the "Long Short Term Memory (LSTM)". Recently, deep learning has been applied in various domains of TS analysis, like classification, forecasting, and AD. Classification and forecasting can be considered fundamental tools to perform AD [298]. I briefly overview the most common TS deep-learning based analysis.

2.6.1 Deep learning based classification

CNNs are the most commonly used structures in classification related tasks. One of the main challenges is how to find the best classifier features, which greatly influences the accuracy and performance of the final model. Historically, manually engineered features by domain experts were preferable given the shortcomings of the available feature engineering methods. However, CNNs

overcame this challenge by learning and extracting important features without need for manual intervention, thereby mitigating the need for human experts [170]. One of the common approaches in handling and classifying TS is flattening data over the time dimension and considering each datapoint as one feature, which generates a very high-dimensional dataset. The intrinsic convolution-based feature selection of CNNs enables this approach to handle a very high-dimensional flattened series. For example, CNN has been applied for the classification of audio signals [173]. Others [5] have used features learned by CNNs as an input for hidden Markovian models, resulting in an improved error rate. However, most applications of CNNs in TS classification are based on univariate TS, which does not apply to real-world contexts. Some have proposed a specific structure to solve this problem [296]. Another drawback to CNNs is the independency of features, especially in relation to flattened TS. Tiled CNNs [280] is one technique that addresses this problem, which is primarily trained with a variety of independent Component (IC) calculation techniques. This structure forgoes the previous assumption that each component is statistically independent and attempts to find a topographic order between them [137]. Two different CNN-based TS classifications include multi-scale convolutional neural network (MCNN) and image processing-based TS classification.

Multi-scale convolutional neural network (MCNN)

This framework improves TS classification by representing features in different time scales and handling noises [62], and includes three sequential stages (Figure 2.6). (1) The transformation stage applies various transformations on the input TS, such as identity mapping; down-sampling transformations; and spectral transformations in the frequency domain, and feeds the CNN with these branches; (2) The local convolution stage applies several parallel convolutional layers to extract features of each branch, separately; (3) The full convolution stage concatenates all extracted features and applies several more convolutional-max pooling layers, fully connected layers, and a softmax layer to generate the final output.

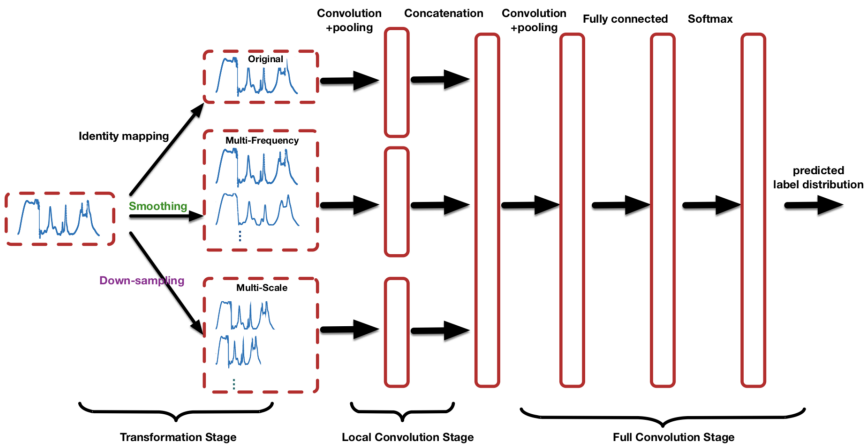


Figure 2.6: Overall architecture of MCNN (adapted from [62])

Image processing-based TS classification

Another classification method proposes transforming TS into an image and applying CNNs to classify the TS image [280], given CNNs have proven capabilities in image processing with a very high level of accuracy. Two suggested methods are: 1) Gramian Angular Field (GAF), and 2) Markov Transition Field (MTF). These transformations, based on the size of TS, may generate prohibitively large images and thus strategies are required to reduce image size without losing too much information. A two-channel image is obtained by combining both image processing methods for classification, which outperforms learning from each individual image. However, this method is only applicable to one-dimensional TS. Other drawbacks of this technique relates to binning the TS values are: 1) Determining the optimal size of bins can be challenging, as increasing the size measure may result in a loss of detailed information while decreasing the size of the bins results in massive images; 2) Very large TS values can be aggregated, or broken down into smaller windows. Moreover, handling non-stationary cases is impossible; and it is not very interpretable for end-users.

2.6.2 Deep learning based forecasting

Recently, various deep-learning methods have emerged for TS modeling and forecasting, several of which are very popular and repeatedly used in this context. For example, RNNs are effective structures in TS modeling and prediction, given their abilities to adapt the backpropagation algorithm to unfold the network through time [241]. Additionally, RNNs keep the memory of the previous inputs by limiting how learned weights change over the training process; however, a major drawback of RNNs is the vanishing gradient [26]. LSTM introduces a complex block of computing units with specific input gates to trap errors without losing the sequential properties of input data [132]. Autoencoder is also a common structure for learning TS features. Stacked denoising autoencoders (SDA) have been used to predict indoor temperature [238], and stacked autoencoders have been utilized to predict the flow of traffic and weather conditions [185, 191]. Deep belief networks (DBN) along with restricted boltzman machines (RBM), are other NN structures that work as a generative graphical model. These include several layers of connected latent variables without any connection inside each hidden layer, and have recently obtained promising results in weather forecasting [118].

Undecimated fully convolutional neural networks (UFCNN) are another variation of CNNs, which have been used for TS forecasting [203]. Some argue fully convolutional networks (FCN) architectures are a type of wavelet transforms simulation and are not robust enough for small translations in the input signals. To overcome this issue, they propose UFCNN as undecimated wavelet transforms are translation invariant. The aim of UFCNN is to substitue both the upsampling and pooling operators from the FCN architecture with upsample filters at the lth resolution level by a factor of $2^l - 1$ through the time dimension.

DeepMinds WaveNet is another breakthrough in TS forecasting, demonstrating the ability of convolutions to capture recurring patterns like specific autocorrelation structures or seasonality [33, 274]. Their convolutional architecture effectively predicts the next observation in a TS, and has

outperformed state-of-the-art recursive networks. This ability comes from dilated causal convolution layers, are able to track temporal aspects even in presence of very long-term dependency (Figure 2.7). Dilated convolutions enable receptive fields, which increase exponentially based on the depth of the convolution layer. Instead of applying causal filters on all points in the sequence, it skips a constant dilation rate of under-process inputs. WaveNet architecture takes advantage of the efficiency of convolutional layers. Moreover, this solves the problem of long-term dependencies over very long time steps, which is one of the long-standing challenges, even for RNNs and LSTMs.

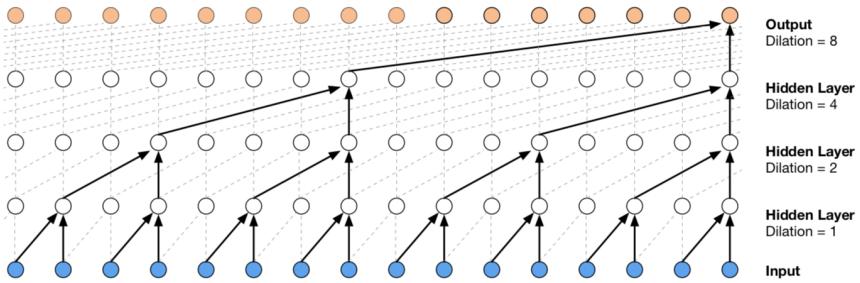


Figure 2.7: A stack of dilated causal convolutional layers (adapted from [274])

2.6.3 Deep anomaly detection

While studies that directly apply deep learning techniques for AD in TS data are not abundant, AD can be performed indirectly based on other main tasks (e.g., classification [171], and prediction [196]). In the second approach, the applied method models the TS behaviour and detects dissimilar regions of data with predicted values as anomalies. Stacked Denoising Autoencoder (SDA) [90] is one such approach, which feeds raw trajectories into a SDA to generate a less noisy, but more robust and meaningful representation of trajectories to feed one-class SVM and detect anomalies.

2.6.4 Reinforcement learning for anomaly detection

Reinforcement Learning (RL) is another type of machine learning models that learns through the exploration and exploitation of an agent or trainer. RL agents' strategies work on the basis of executing actions and observing the feedback from the environment in the form of positive/negative rewards. After receiving a reward according to the performed action, the agent observes any changes in the environment and updates its policy in order to maximize the future rewards [267].

The AD problem can be formulated as a reinforcement learning problem, where an autonomous agent interacts with the environment and takes actions (such as allowing or denying access) and gets rewards from the environment (positive or negative rewards for correct or wrong predictions of an anomaly, respectively) and over a process learns to predict anomalies with a high level of accuracy [286]. Currently, there are a few studies that utilize RL to address AD challenges. For example, [188] applies forward RL to detect anomalies and [70] utilizes inexplicability scores to

find goal-directed trajectories. Furthermore, [215] proposes sequential anomaly detection using inverse reinforcement learning (IRL) in order to determine the decision-making agents underlying function which triggers his/her behavior. The agents normal behavior will be understood by the reward function, inferred via IRL. Also, a deep reinforcement-learning-based approach that seeks novel classes of anomaly beyond the labeled training data is proposed in [220].

It is worth noting, most of the designed RL-based AD methods still require the predefined notion of reward signals. Also, depending on the feedback loop and reward establishments, they may need more than the current data item in order to define states. In general, RL is a great solution to problems that require making decisions that influence the world; for example, RL-based AD helps where the target agent is intentionally malicious or goal-directed [215]. Also, RL may be more useful in hyperparameter learning or data collection rather than deciding which observation is normal or anomalous.

2.6.5 Hybrid methods

Ensemble and hybrid methods utilize the information fusion concept in slightly different ways. Ensemble models combine multiple but homogeneous, weak models, and typically apply various merging methods at the level of their individual output [147]. Hybrid methods primarily combine heterogeneous models and various machine learning approaches. Both techniques can lead to considerably higher accuracy and perform ace solutions. Individual methods are grouped together because the existing models:

- Lack sufficient data to represent data distribution;
- Provide locally optimal results due to the dependency on starting points;
- Cannot be modeled based on a single hypothesis given the functionality of the data, which is better approximated by a weighted sum of several datasets.

In the TS analysis context there are multiple studies and frameworks that can be categorized as an ensemble or hybrid method. Among these approaches, several have applied very interesting logic to combine model-driven and data-driven methods to cover the potential weaknesses of both. Others have considered ANNs as suitable candidates for a data-driven mode; for example, ANN has been combined with ARIMA for TS [152], water quality [89], and photo-voltaic power generators forecasting [80], respectively. Another hybrid method combines SVM, ensemble empirical mode decomposition, and partial autocorrelation function to forecast the speed of the wind [135], while a hybrid fuzzy model takes advantage of the Fuzzy C-means method for fuzzification and ANN for defuzzification [80].

2.7 Summary

In this report, a variety of TS analysis algorithms in the context of TS based anomaly detection are reviewed. To this end, I have categorized them based on the main paradigm and highlighted their main

features and approaches. A summery of the reviewed algorithms and their specific characteristics is presented in Tables 2.2, 2.3.

Notwithstanding having a very rich history, TS modeling and anomaly detection is still an ongoing topic in machine learning research. Nowadays, embedded control systems for operating critical infrastructure interact with their physical environment by reading data signals and generating control signals in response. Hence, latent stochastic, or semi-stochastic, external factors influence the system behavior and complicate such predictive problems. Reviewing the existing works provided us with enough insight to discover the existing challenges and promising directions for future research to improve the state of the art approaches.

Original task	**Approach**	**Strategy**	**Pros**	**Cons**
Direct anomaly detection	Unsupervised discriminative methods	Cluster available sequences to find rare cases	Label is not required	highly dependent on similarity evaluation function; not applicable to stream data
	Unsupervised parametric methods	Isolates rare generation probabilities	Label is not required	Not flexible; not applicable to stream data
	Supervised methods	Classifies data based on their data	All of the unbalanced classifiers are applicable	Label is required (not practical)
Window-based anomaly detection	Normal pattern repository	Rates sequences based on the historical normal observations	Finds localized anomalies	What is proper window size?; needs discretization
	Negative patterns repository	Keeps track of anomalous patterns to rate the new observations	Needs smaller repository	Not able to detect zero-day attacks; what is proper window size?; needs discretization
	Mixed patterns repository	Keeps track of both normal and anomalous cases	Takes advantage of both repositories	Higher space-complexity; needs discretization
Subsequence-based anomaly detection	Tracking all the possible subsequences	Finds rare subsequences utilizing pattern matching	Keeps track of all the possible sizes in the observed subsequences	Not applicable to non-stationary cases; needs discretization

Table 2.2: Table of summary - Anomaly detection in a time series database

Original task	Approach	Strategy	Pros	Cons
Point anomaly detection	Prediction-based methods	Computes anomaly scores based on deviation from the expectation	Strong prediction methods	Difficulties in distinguishing noise from anomalies?
	Profiling-based methods	Computes anomaly scores based on the variance vector of comparing a new point vs normal profile		Not applicable to non-stationary TS; needs discretization
	Deviant-based methods	Finds the points that the TS histogram's error-bound decreases in their absence	Able to find anomalous points in the middle of a sequence	Not applicable to non-stationary data; need discretization
Sequence anomaly detection	Discord discovery	Finds the subsequences with the largest distance to their non-overlapping subsequences	Able to find variable-size anomalies	Expensive in big data; not applicable to non-stationary data; need discretization
Multi-scale anomaly detection	-	Compares obtained patterns of subsequent points vs the other patterns	Applicable to irregular time series	Needs discretized sequences

Table 2.3: Table of summary - Anomaly detection in a time series

Chapter 3

An Anomaly Detection Algorithm

The primary purpose of this chapter is to systematically identify and analyze possible vulnerabilities as a step toward enhanced risk awareness and management. In other words, the study addresses situational awareness and the proactive side of cybersecurity, rather than the reactive side dealing with emergency response and digital forensics to mitigate the impact of attacks and identify the source [51]. Cyber situational awareness and assessment as discussed here are based on Big Data analytics used for anomaly detection, specifically methods for automatic detection of suspicious abnormal behavior pointing to a cybersecurity breach or other illegal activities. Intuitively, anomalies are isolated events in data that do not conform to expected or normal behavior. Given some set of observations, the problem of anomaly detection is that of separating a small minority of anomalous data from a large majority of data. In the study of cybersecurity, the more sophisticated task is differentiating *suspicious* anomalous behavior (any behavior that points to a potential threat to cybersecurity) as anomalies of interest from the set of anomalies considered irrelevant to security.

Hidden Markov model (HMM) family are well-established and interpretable methods for modeling and recognizing temporal patterns as well as differentiating between normal and anomalous behavior of a target system [269]—especially in cases where one needs to identify the unobservable state of the system based on its external observable characteristics. HMM-based anomaly detection methods are mostly evaluated based on likelihood measuring factors or entropy [141, 269], which summarizes and represents the overall observable evidences (i.e., observations). However, in some real-world problems, it is crucial to put a magnifier on the observation sequence for the purpose of situation analysis.

In this chapter, I propose a novel methodical framework, called AnomalyTracker, which learns the normal behavior of a water supply system to trace anomaly scores for each data point. The proposed anomaly detection framework aims at advancing the state-of-the-art in temporal anomaly detection by making the following contributions: *1)* Proposing a multi-granular anomaly detection approach by applying a hierarchical model with the possibility of detecting anomalies in different sizes of test-windows. This approach, not only does enhance the true positive ratio but also reduces the false negative ratio. *2)* Using an improved HSMM-based anomaly detection method with the possibility of detecting the exact anomaly point in a given data sequence. The proposed approach outperforms

HSMM-based anomaly detection methods which use log-likelihood or entropy measures [141, 269] or the number of mismatches [281]. *3)* Addressing collective and contextual anomalies of the system as opposed to point-based anomalies [51] which are generally addressed in the literature [208].

3.1 Problem definition

This project is focused on proposing an efficient and interpretable anomaly detection algorithm to trace suspicious activities in supervisory control systems of critical infrastructures. The considered system's behavior is monitored over some time interval T, comprising t consecutive time steps, forms a multivariate time series $X_T = \langle x_t : t \in T \rangle$, $T \subseteq \mathbb{N}$. Intuitively, each element of this time series is a multivariate process with m observed features ($x_t \in \mathbb{R}^m$), meaning $x_i = (x_i^1, x_i^2, ..., x_i^m)$. Thus, an online and interpretable anomaly detection framework (ADF) using *Hidden Markov Model (HMM)* is desired to spot any suspicious anomalous event in near real time by continuously analyzing the stream data. The goal is to distinguish the anomalous events, which likely correspond to a cyberattack.

$$ADF(x_{t+1}|X_T) \rightarrow \{Normal, Anomaly\} \tag{3.1}$$

3.2 Background

This section discusses published work on anomaly detection in supervisory control systems and HMM-based anomaly detection.

3.2.1 Anomaly detection in supervisory control systems

Nowadays, critical infrastructure widely used for municipal water management, energy supply, oil and gas pipelines, public transportation and beyond typically depends on computer networks and automated control systems to operate and monitor routine processes. Supervisory control systems or Industrial Control Systems (ICS), such as Supervisory Control and Data Acquisition systems (SCADA), improve the services and their reliability. However, this reliance on ICS leaves critical infrastructure potentially vulnerable to targeted cyberattacks or accidental cyber events [7]. Any interruption in water management services caused by a cyber or physical attack could erode public confidence or, even worse, have significant public health or economic consequences [7, 117]. Particularly, in water management services, supervisory control systems generally control various equipment and monitor water transport, distribution, and treatment. Analog and digital input and output modules track and measure different attributes including water levels in large reservoirs, tanks, water pressure, and flow in pipes [117].

Improving the security of supervisory control systems is not a new topic but an important one. A large body of literature exists on enhancing SCADA safety and security to prevent physical destruction as well as economic losses and threats to humans. Most of these studies are related to network intrusion detection systems using supervised approaches [116]. These methods are able to

detect attacks with simple mechanisms already known before, while detecting new malicious actions is not easy for such methods. A major drawback of intrusion detection-based methods is that they require prior knowledge of different types of attacks for efficient anomaly detection.

Another category of works related to the security of SCADA systems is anomaly detection approaches using machine learning methods [208]. In such methods, the periodicity of SCADA traffic is used to define normal behavior and detect attacks to SCADA protocols as anomalous behavior. Hidden Markov Models (HMMs) and their variants are one of the popular machine learning approaches extensively used for anomaly detection in time series. In the next section, I briefly study HMM-based anomaly detection approaches.

3.2.2 HMM-based anomaly detection

An HMM is a probabilistic model in which the target system is represented as a Markov process with hidden states. Unlike the simple Markov model, in a hidden Markov model the state is not explicitly visible, while the output depends on the state. As expressed in [226], "An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols."

An HMM is specified by a triple $\theta = (\pi, P, b)$, where π is the distribution of the initial state, P is the transition matrix of the underlying Markov chain, and b is the emission distribution—conditional distribution of observed variables given the unobserved (or hidden) states [59]. In some real-world problems, it is unrealistic to assume the sojourn time, or mean waiting time, is geometrically distributed but instead the probability of a state change depends on the time spent in the current state. A possible solution is to estimate the duration density d which produces a Hidden Semi Markov Model (HSMM). An HSMM is specified by a quadruple $\theta = (\pi, P, b, d)$ where π is the distribution of the initial state, P is the transition matrix of the underlying Markov chain, b is the emission distribution—conditional distribution of observed variables given the unobserved (or hidden) states and d estimates the duration density [59].

HMM is one of the most widely used traditional approaches for anomaly detection and applied across different domains. Using HMM for intrusion detection has been first proposed by Warrender *et al.* in [281], in which they propose an HMM-based approach to detect anomalous program traces in operating system calls data. In this work, an HMM is trained using the training sequences and tested using two methods. In the first method, the likelihood of a test sequence generated by the learned HMM is computed using the Viterbi algorithm. In the second method, the underlying Finite State Automaton (FSA) of the HMM is used to count the number of times that the HMM makes an unlikely state transition or outputs a mismatch. Then, the number of mismatches indicates the anomaly score of the input sequence. In [294], a hierarchical HMM-based anomaly detection approach is proposed which decides based on the number of mismatches using a second-level HMM.

In [225], the authors build a database of state transitions of normal sequences obtained from a fitted HMM. Given a test sample, the method then compares its state transitions to the normal ones

in the database. The test sample is considered an anomaly if the number of mismatches exceeds a threshold. In [141], the normal behavior of user browsing and network-wide traffic is modeled using HSMM to detect rare sequences. The authors of [202] apply HSMM in the same problem domain to detect anomalies based on the likelihood difference of every sequence in comparison with normal behavior.

In this work, I present an HSMM-based multi-granular anomaly detection approach using a hierarchical model. The closest group of methods to this work, particularly in regard to computing the anomaly score based on predicted FSA, are two basic methods presented in [225, 281]. Unlike [225], which maintains a repository of normal sequence of states to evaluate the anomaly score of test windows regardless of contextual information, my method creates an online normal sequence of states based on the context of test windows in order to test and identify their anomaly level. Therefore, my method is stronger in detecting contextual anomalies as discussed in Section 3.4.2.

Similar to [281], my method evaluates the anomaly score of each data point using FSA. However, as I explain in Section 3.4.2, due to specific properties of the decoding phase, the number of mismatches cannot be a reliable criterion for detecting the actual anomaly score. This issue can be addressed by evaluating the anomaly score of each point individually and switching to regression-based anomaly detection in cases where the HMM cannot follow the real trend of the data. In the studies related to anomaly detection in supervisory control systems data, [208] addresses the same problem as I do and propose an SVM-based classification approach for this purpose. In [208], a labeled dataset is used, which is not the case in this domain and usually leading us to apply unsupervised or semi-supervised approaches. The method proposed in [208] ignores the temporal aspect of data affecting its performance and usability in detecting contextual anomalies.

3.3 Addressed Challenges

Generally speaking, most of the previously proposed HMM-based techniques suffer three major limitations, which have been addressed in this study.

First, to score and find the anomalies, most of the related studies are using measuring factors such as log-likelihood [141], entropy [202]. While some others are using the frequency of mismatches [281] or a database of normal state transitions [225]. However, there exist some types of anomalous scenarios that cannot be efficiently detected using the mentioned factors (see Section 3.4). Particularly, in adversarial settings, adversaries try to blend in with the distribution of normal points [82]. When the attacks (targets of interest) are not confined to extreme anomalies, or when the extreme anomalies are not anomalies, like swamping phenomena, the anomalies of interest will be confused with normal points or with uninteresting anomalies. So, only relying on the sequence-based likelihood to find the anomalous/suspicious activities would be inefficient to detect attacks.

Second, none of the existing HMM-based techniques can address point-wise anomaly detection or specify the exact point of an anomaly. Likelihood-based techniques give an idea about the overall

condition of the test window, while sometimes it is very valuable to locate the exact timestamp of anomalous events in order to identify the criticality level.

Third, finding the right size and granularity for the input window is a very fundamental concept and highly challenging in statistical and ML models. The input window should be long enough to be informative and representative of a system's behavior; meaning that a window covering all of the change-points provides more knowledge. However, various methods have different levels of capacity; sometimes, more information may misguide a method and diminish the role of closer intervals. Also, the granularity level of reconstructed trajectories matters. A very fine granular input may decrease the performance, whereas a coarse-grained sampling rate may limit the real-world practicability. The proposed method solves these challenges by utilizing a hierarchical model with the possibility of detecting anomalies in different sizes of test-windows.

3.4 Proposed Method

AnomalyTracker is proposed to: *1)* learn the behavior of water stations from historical data; *2)* use the extracted patterns to detect anomalies in the current time; and, *3)* predict the future state of water stations. The high-level overview of AnomalyTracker, which comprises four main components (see also Figure 3.1), is followed by the definitions of *test window* and *reference window*.

Basic Definitions:

- *Test Window (TW)* is defined as a period of time which is under investigation to detect any possible anomaly.
- *Reference Window (RW)* is defined as the most *recent* genuine (i.e., normal) window to a *TW*, which is also in approximately the same context (e.g., temperature and time interval) of the *TW*.

Approach Overview:

- *Behavior Tracking*: This step constructs a hierarchy of Markovian models representing various-length baseline bahviour of the system based on historic data to be used for the purpose of anomaly detection in the test data.
- *Threshold-based Filtering*: This step applies a rule-based method on the test data to filter out the outliers–values in the time series which are out of the range of predefined thresholds.
- *Anomalous Pattern Matching*: Given a TW, a set of test sequences is constructed by extending the TW using its various-length prefixes. This way, TW can be examined against existing various-length anomalous or misuse pattern profiles. If any of the test sequences matches with an existing profile, TW will also be flagged as suspicious and ranked based on the critically level of the matching pattern.

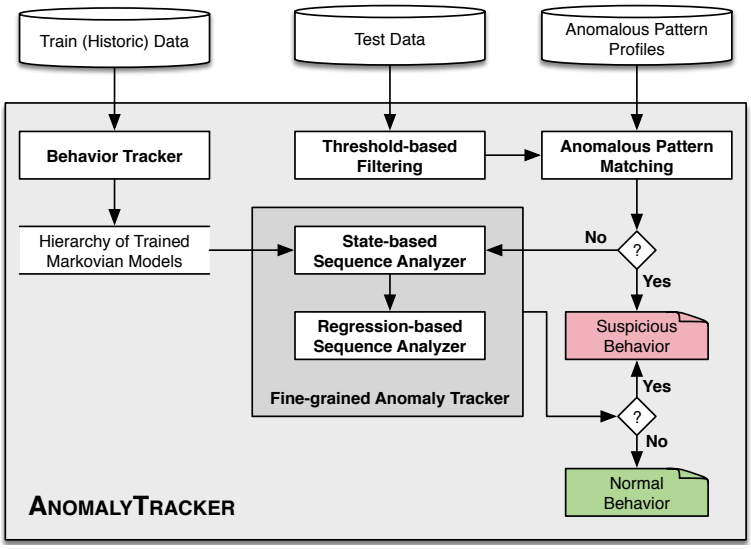


Figure 3.1: Control Flow between Analytic Components

- *Fine-grained Anomaly Tracking*: If none of test sequences of TW matches with any anomalous profile, then it is required to analyze TW with the baseline behavior, represented in the hierarchy of Markovian models, for detecting any possible unknown anomalies. This step measures and compares the distance between a given TW and a set of RWs to decide on the existence of any possible anomaly and its critically level.

3.4.1 Behavior tracker

The state of water stations changes with meaningful patterns. Analyzing water stations states shows long-range dependency property for different features of water stations. For instance, one can see such dependency in the scale of several minutes for the flow level feature.

Since, there is a high dependency between the values of consequent observations in the time series of the water station states I use HMM to model the behavior of water stations. On the other hand, the state duration distribution is not uniform making it difficult for HMM to model water stations behavior. Among different variations of HMM, I specifically employ HSMMs, which is able to successfully address the long-range dependency phenomena and model state duration in the continuous time series which is a characteristic of water station states.

The proposed behavior tracker approach customizes HSMMs to overcome challenges related to the water stations characteristics, including *seasonality effect* and *irregularity*, and learn their behavior more efficiently. These two issues and the proposed solutions are described below.

Irregularity. The water stations data is distributed unevenly, meaning that timestamps are not periodic, which results in so-called *irregular time series*. Standard HSMM is not able to handle

such time series because they rely on the assumption that observations are recorded in evenly time intervals [101]. However, in many real-world applications, capturing the observations in exact timestamps is not feasible. To address this issue, I propose a different approach by adding a new derived feature—time difference between every two consecutive observations—and training the model with the new multivariate time series data. There is a latent correlation between the time difference feature and original features such as flow level which contributes to HSMM to overcome the irregularity aspect.

Seasonality Effect. The *seasonality effect* is about occurrence of any periodic variation in the time series data [79]. For instance, the covariance between temperature and flow level confirms the seasonality pattern that flow level trends peak during the summer and decline until the winter. We can see other seasonality patterns related to people's lifestyle as described in the next section. It is discussed in the literature that there is no established predictive method to be continuously successful for a long period of time [113].

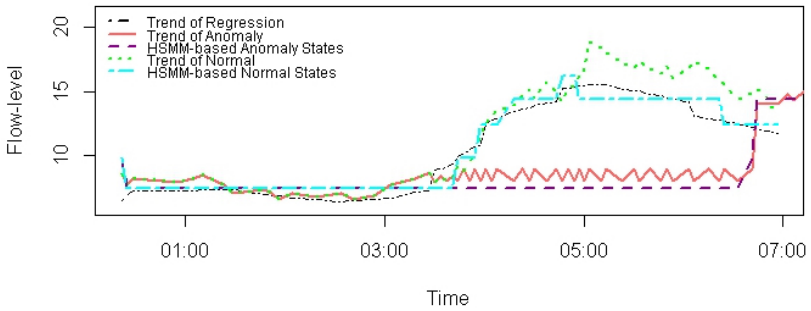
To address this issue, I propose using an agglomerative hierarchical approach. Starting from a short time interval which includes more similar data points, I train an HSMM representing the water station behavior for that time interval. Then we should merge time intervals to larger and larger intervals until all data points are in a single sequence. During the merging procedure, one is constructed in each step for each time interval. This way we are able to extract the patterns in different time intervals which might be completely different than the overall pattern.

The HSMMs learned from shorter time intervals are located in lower layers of the hierarchy. They include more detailed description of the water stations behavior but are very context sensitive. On the contrary, the HSMMs learned from longer time intervals are located in upper levels of the hierarchy. They have a more global view of water stations behavior and do not loose the information in transition between shorter time intervals; however, their performance is easily affected by the seasonality patterns. Combining HSMMs learned from shorter and longer time windows resolves the overfitting/underfitting problem.

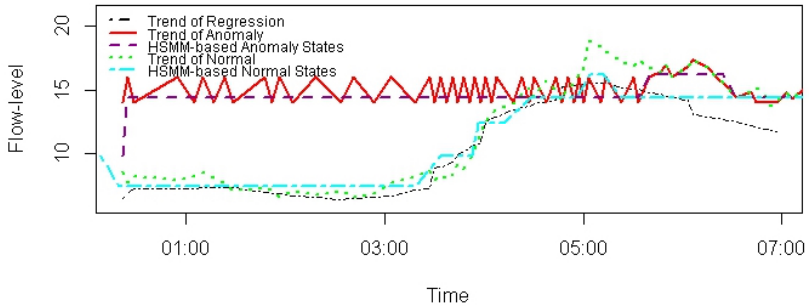
3.4.2 Fine-grained anomaly tracker

Markovian models are well-established for anomaly detection in different real-world domains. In this section, I propose an approach to detect anomalies of interests and infer anomaly score using the combination of Markovian (e.g., HMM and HSMM) and regression models. Viterbi decoding algorithm is utilized to define a metric for measuring the criticality level of detected anomalies. Viterbi is a dynamic programming algorithm that recursively generates the most likely sequence of hidden states for a given sequence of observations using the optimality [24].

In many real-world scenarios, the decoded states related to a normal observation sequence is expected to follow the time series trend with ignorable differences; however, this statement is not always valid—especially in cases that the divergence of observed normal sequence are high with



(a)



(b)

Figure 3.2: Drawbacks of Log-likelihood and Mismatch Counting in HMM-based anomaly detection

respect to the baseline model. Many HMM-based anomaly detection methods are using measuring factors such as log-likelihood [141], entropy [202], and counting the number of mismatches [281]; while there exists some types of anomalous scenarios that cannot be efficiently detected using such measuring factors.

Let's briefly exemplify two scenarios highlighting the weakness of the log-likelihood measuring factor. Figure 3.2a illustrates the first eight hours of a day, in which the normalized negative log-likelihood of the fitted model is 1.61 with the standard deviation of 0.2 for about 80% of observations—meaning the observation values between 1.41 to 1.81 are normal. The normal trend is shown by green dotted line with the average negative log-likelihood of 1.501 with one improbable emission. The predicted state sequence is shown by blue dashed line. Red line shows a possible anomalous scenario which starts before 4:00 AM and ends 10 minutes before 7:00 AM. The Viterbi algorithm decodes the red sequence with order of states shown in purple with the negative log-likelihood of 1.56 and one improbable transition after the anomaly interval. This example illustrates that some anomalous cases do not change the likelihood (entropy) in a way that it helps for anomaly detection. In Figure 3.2b, the Viterbi does not detect any improbable transition or emission and the normalized negative log-likelihood is 1.68 which misleads toward a normal window. Weaknesses of the likelihood measuring factor are:

- On the one hand, some anomalous scenarios do not change the likelihood much—particularly when some parts of the scenario can be seen in the overall normal behavior; on the other hand, likelihood is highly sensitive to very rare observations which can also be a noise.

- Likelihood gives an idea about the overall condition of the test window, while sometimes it is very valuable to locate the exact time stamp of anomalous events in order to identify the criticality level.

Therefore, I propose a novel method, called *state-based sequence analyzer* to investigate a TW based on related RWs. Considering the context (i.e., *flow level* and *temperature*) of TW, the closest model in the leaf nodes of the hierarchy is selected to decode the most likely state sequence of TW using Viterbi decoder. Since patterns in real-world data are not identical, in order to increase the confidence factor, it is essential to expand TW with (various lengths) prefixes to also use the Markovian models in the higher levels of the hierarchy. Based on the context of TW, one of the higher level models is used to generate the sequence of states corresponding to the observations in the larger window, as well as the overall sequence of transitions of the test window TW and all of the reference windows RW_i.

State-based sequence analyzer. The behavior tracker is expected to follow the overall trend of data in the RWs. This phase decodes and compares the most probable state sequences of RWs and TW based on the hierarchy of Markovian models to detect possible anomalies of interest.

Step 1. Since trained Markovian models can diverge from the real-world overtime, we first need to verify that these models are able to fit genuine normal RWs. Therefore, we should reconstruct the observation sequence of each RW, called *decoded sequence*, using the Viterbi algorithm and the emission matrices, and compare the corresponding observations in real and decoded sequences. Then, the number of mismatched cases in which the difference between the real and decoded observation is greater than a threshold is counted and the sequence is marked as *unfitted* if the number of mismatched cases is higher than an expected value.

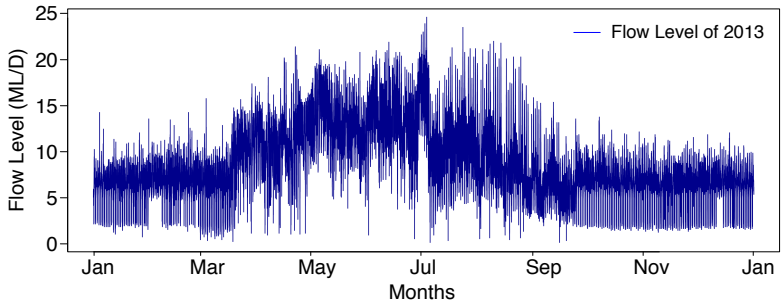
Step 2 If the weighted rate of unfitted RWs is higher than expected—meaning that the Markovian models are not up-to-date and need to be retrained—then the *regression-based sequence tracker* should be used to compare TW with RWs; otherwise, the algorithm proceeds to the next step.

Step 3 For each sequence RW_k in the reference group, first the fitted sequence of states on TW with the predicted ones for RW_k is computed to calculate the distance between TW and the RWs as follows:

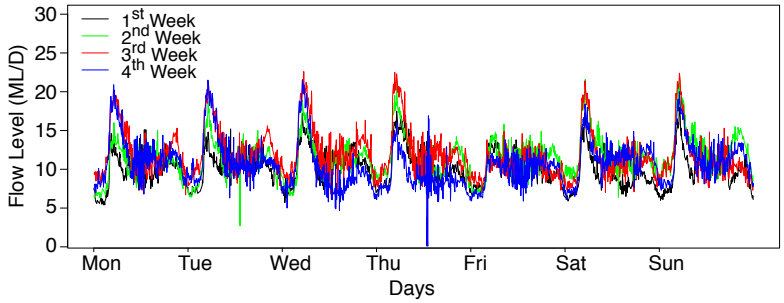
$$\forall o_i \in TW : d_{i_k} = |o_i - x_n| \quad \text{where: } x_n \in RW_k \ \wedge \ n_{j=T_1^k:T_M^k} \left|(\mathcal{T}_{x_j} - \mathcal{T}_{o_i}) - (T_1 - T_1^k)\right| \tag{3.2}$$

T_1^k and T_M^k are the minimum and maximum timestamps in RW_k. T_1 and T_M are the minimum and maximum timestamps in TW. Then mismatch observations should be calculated as follows:

$$d_{i_k} \geq confidence \times (\sigma_{s_{o_i}} + \sigma_{s_{x_j}}) \Rightarrow map_{i_k} = false \tag{3.3}$$



(a)



(b)

Figure 3.3: (a) Water level for the year of 2013; (b)Water level for the days of weeks of July 2013.

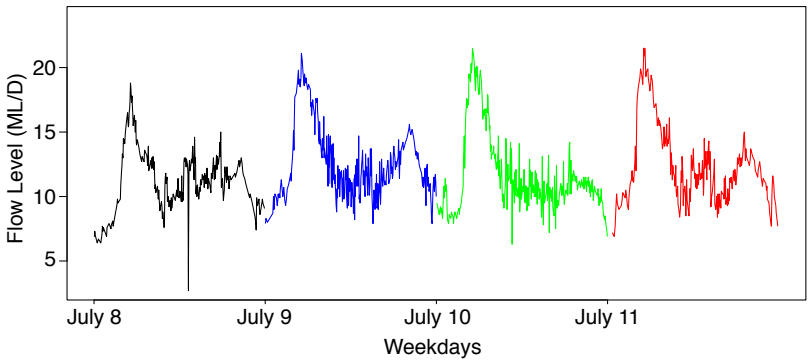
The number of false values of map_{i_k} shows the mismatch rate which is used in reporting the anomaly score of *TW* sub-windows.

Regression-based sequence analyzer. This phase constructs and trains a weighted regression model on RWs. The difference of each observation of TW with the regression is calculated to compute the Least Square Error (LSE) of TW. If this error is higher than the expectation with respect to the training data, then TW is reported as anomaly with highlighting its main anomalous sub-windows.

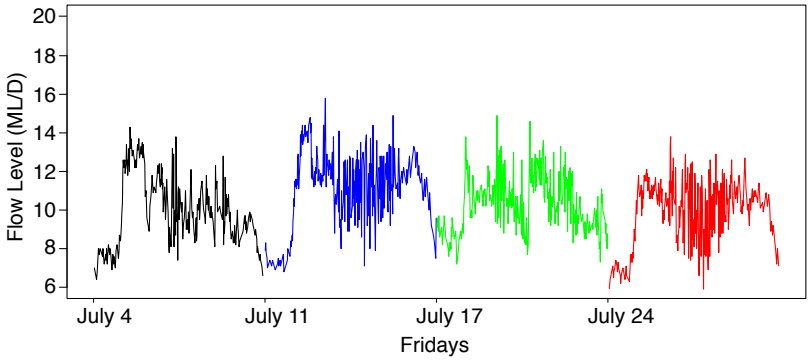
3.5 Dataset Characteristics

The set of experiments are performed based on a dataset from the SCADA-based water supply system of the City of Surrey in BC, Canada for the time period of 2011–2014. This dataset contains the historical log-files of various water stations including drinking water, drainage, sanitary and vacuum stations. For this study, I use the data of water stations in the form of multivariate time series for the mentioned time period. For each water station, at a specific time, there are several features including inflow, outflow, water level, temperature, and running status.

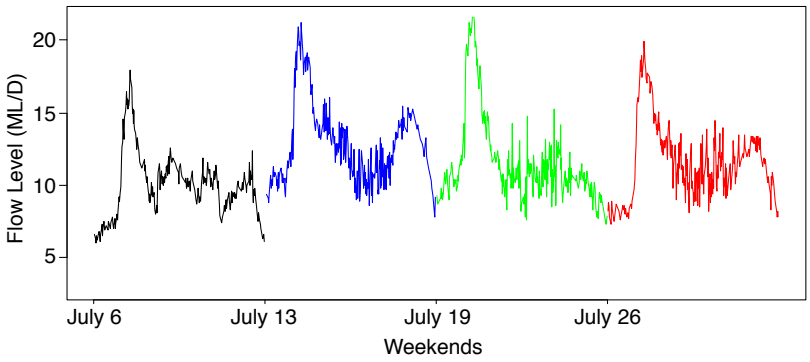
Seasonal fluctuations can be a major reason for variations in residential water consumption. Through analyzing the dataset, I determined the seasonality patterns at various levels of aggregation, which contributes to detecting irregular patterns more efficiently. One can observe daily, weekly and monthly as well as seasonally strong patterns in the water consumption.



(a)

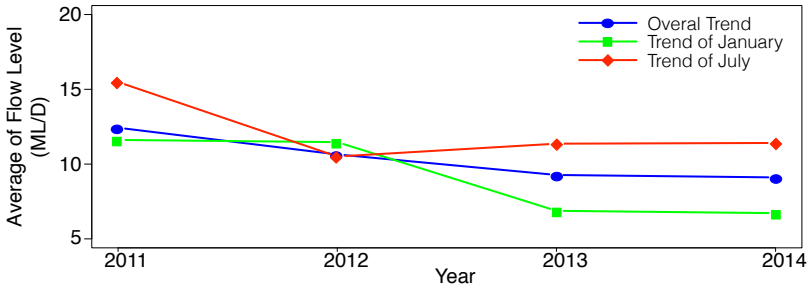


(b)

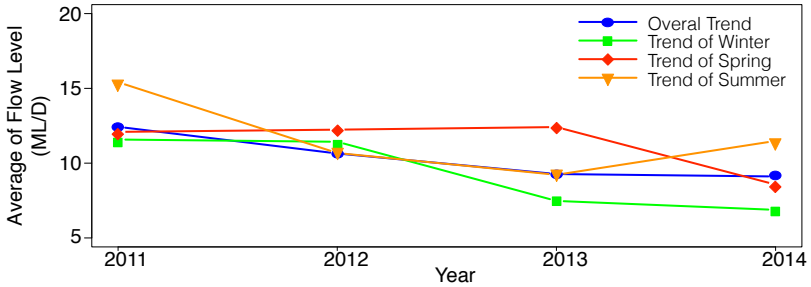


(c)

Figure 3.4: (a) Water level for four different week days of the month July 2013; (b) Water level for four different Fridays of the month July 2013; (c) Water level for four different weekends of the month July 2013.



(a)



(b)

Figure 3.5: (a) Opposite trend in comparison of water level of months in different years; (b) Opposite trends in comparison of water level of seasons in different years.

For instance, each year can be divided into high-demand and regular-demand seasons. Weather, specifically the outside temperature, is the primary factor in determining the partitioning time. As expected, water demand increases with higher temperatures. As Figure 3.3a shows, the flow level in a high-demand season (May to September) is higher than for the rest of the year. In AnomalyTracker and the related experiments, I divide the training data into high and regular seasons.

Additionally, as Figure 3.3b shows, in the flow level for the four weeks of the month July 2013, one can observe different usage patterns for different days of one week but a similar pattern for the same days in all of the four weeks. Interestingly, there is a different pattern on Fridays compared to the other four weekdays. The weekends have their own specific patterns in terms of the trend and peak times. The mentioned pattern is more obvious in Figures 3.4a, 3.4b and 3.4c, which show the similarity of patterns in three different categories: weekdays, Fridays, weekends, and also the significant difference in comparing the patterns of a category with another one. For instance, the peak times on weekdays are early morning and evening, while this is shifted in the weekends. Moreover, one can observe that the flow level at nighttime is lower than at daytime regardless of the day of the week. AnomalyTracker considers these extracted patterns to fit the best model to data.

Another important pattern is observable when comparing the pattern of changes in two different time intervals. For instance, in Figure 3.5a, the flow level in July 2013 exceeds the one in July 2012, while we see an opposite pattern for the month of January, where the flow level in January 2013 is less than in January 2012. A similar phenomenon is depicted in Figure 3.5b, where the trends of the flow level for summer and winter seasons are different. Therefore, simply deciding based on the

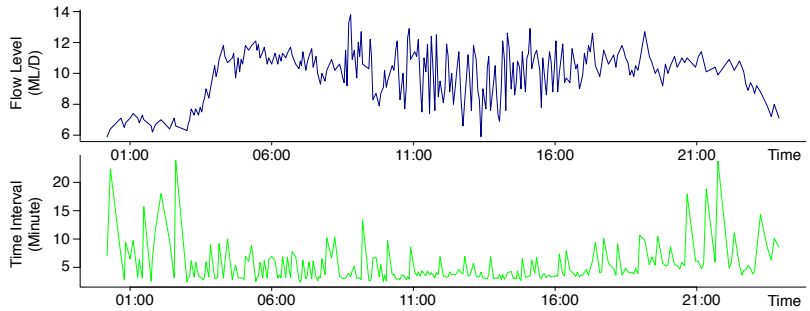


Figure 3.6: Time difference with respect to the water level in a sample day (July 25th)

overall trend of the flow level over years is not enough but rather we need to consider the temporal context.

Time series data are typically assumed to occur at regular time intervals. But the data used in this study is irregular, meaning that the measurement of water station features did not happen at a regular time interval. We can see this irregular pattern in Figure 3.6, where the water flow level and time difference between each pair of consecutive data points is shown. On the other hand, Figure 3.7a displays the periodic pattern of time difference over four years that indicate hidden patterns in time difference sequences. Moreover, Figure 3.7b at a finer level shows that the distribution of time intervals over a regular-demand season is more scattered than over a high-demand season.

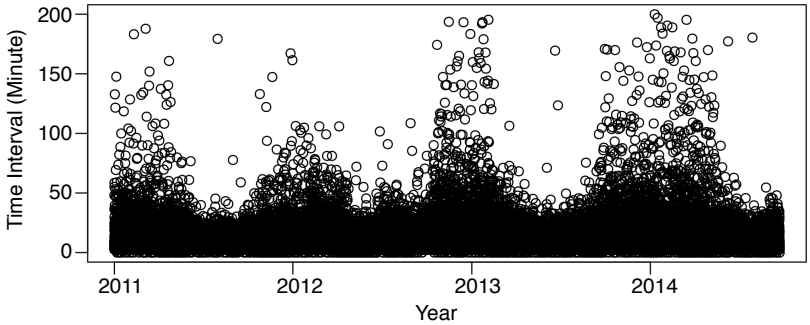
3.6 Experimental Evaluation

This section presents the experimental results for evaluating the proposed method in comparison with the baseline methods. To evaluate the performance of anomaly detection methods, three measures, *precision*, *recall*, and *F-measure* are used.

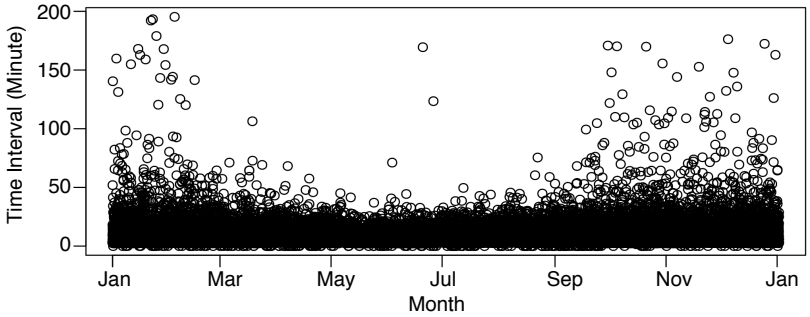
3.6.1 Experiments setup

In this study, I use records of the first three years for training the anomaly detection model and the last year to generate the test data. The used dataset is unlabeled and not including anomalous cases confirmed by the domain experts. This set of experiments use three main features of water stations including water flow level, temperature of water station, and time difference between each pair of consecutive data points denoted by FlowLevel, Temperature and TimeDifference, respectively. The test samples are generated in three ways:

Anomalous Samples. For evaluating the performance of AnomalyTracker, a set of possible anomaly scenarios should be defined. For each scenario, the anomalous cases are generated and inserted into the normal data. The augmented data is used as the test data. In collaboration with domain experts, four types of out-of-distribution scenarios are identified:



(a)



(b)

Figure 3.7: (a) Time difference of data points for the years 2011-2014; (b) Time difference of data points for the years 2013.

- *Maximum Flow.* Every water station can produce a range of flow and pressure. Water flow exceeding the maximum flow capacity should be considered as an anomaly and a critical situation for the water network. To test the AnomalyTracker performance in detecting this type of anomaly, extreme noises with values greater than the maximum flow capacity are added.

- *Minimum Flow.* We are able to learn the minimum flow level of a water station from the historical data. If the water flow level falls below the predefined minimum flow level, it is an anomaly that can risk water network operation. To experiment on this type, distributed synthetic noises with values smaller than minimum flow levels are added.

- *Continuous Overflow.* Assuming that the flow level can be predicted successfully, we may continuously encounter cases with the real flow level greater than the expected value but less than the maximum flow capacity. Although this type of anomaly is not as harmful as exceeding the maximum capacity, it should be considered as an anomalous case.

- *Frequent Overflow.* This type of anomaly is similar to continuous overflow anomaly but happening intermittently, not continuously.

For each of the above categories, 10000 samples are extracted from the data of the year 2014 and noisy data are added to each of them as representative of anomalous behavior.

Manipulated Samples. For making a real data test sample, I extract a times series of records of the year 2014 (not used for training the model) and then remove a continuous subset of the extracted time series with the size of 100. In other words, a real data sample is a subset of records of the year 2014 not containing a continuous subset of 100 records to generate an abnormal sequence. Although a manipulated sample does not belong to one of the above-mentioned anomaly types, one would expect that a strong model differentiates a manipulated sample from a normal sample. This set includes 1000 different samples.

Noisy Test Data. For more realistic evaluation of HMM-based methods in which log-likelihood is used to detect anomalies— where these methods are not able to identify and differentiate the *type* of anomalies from the mentioned four types—a separate test data, called noisy test data, is generated. Each sample of this test data includes at least 15% contextual noisy points or out of range values representing an anomaly sequence. This test data includes 1000 samples.

3.6.2 Evaluation of HMM-based models

This section compares the performance of variations of HMMs. This experiment helps to select the strongest version of HMMs to be used in AnomalyTracker. Below is the list of evaluated HMM variations:

- **U-HMM.** This is a univariate HMM trained using the FlowLevel feature as the main feature representing the behavior of water supply system.
- **U-HSMM.** This method is a univariate HSMM trained using the FlowLevel feature.
- **M-HMM.** This is a multivariate HMM fitted using FlowLevel, Temperature and TimeDifference features. The Temperature and TimeDifference features are supposed to provide more contextual knowledge about the reasons of seasonality in water flow level.
- **M-HSMM.** This is a multivariate HSMM fitted using FlowLevel, Temperature and TimeDifference features.
- **HM-HMM.** This method is a hierarchical multivariate HMM working on three layers of daily, weekly, monthly time intervals.
- **HM-HSMM.** This is a hierarchical multivariate HSMM working on three layers of daily, weekly, monthly time intervals.

I define two experiments to study the performance of HMM variations:

Experiment 1. HMMS is fitted to the training data and then compare their log-likelihood for a set of test data sequences. The method with the smallest average log-likelihood works best. To perform this experiment, the manipulated samples are used. As presented in Table 3.1, HM-HSMM outperforms the other methods.

Train-based interval approximates the interval of log-likelihood for randomly selected group of train sequences based on the fitted model. The less sparse distribution of log-likelihood shows that the fitted model follows the trend of train data better. As shown in Table 3.1, the train-based interval of HM-HSMM is the densest among all studied methods.

The superior performance of HM-HSMM has three main reasons: *1)* Using HSMM instead of HMM: due to the long-range dependency of data, the statistical distribution of state duration in HSMM works better than explicit self-transition in HMM. As shown in the results, even in the same situation, U-HSMM and M-HSMM outperform U-HMM and M-HMM, respectively. *2)* Applying multivariate data instead of univariate: Temperature and the TimeDifference of observed flow levels are used as contextual information for modeling the behavior water supply system. The more successful performance of multivariate methods comparing to the similar univariate ones shows the latent relation between the mentioned contextual features and FlowLevel feature in the context of time. *3)* Applying hierarchical model: modeling data based on windows in different granularity provides the model with the possibility of fitting better to the data in more related context. The experimental results in Table 3.1 justifies this argument where the HM-HSMM and HM-HMM outperform the other methods.

Table 3.1: Performance of variations of HMM using anomalous samples

Model	**Negative Log likelihood**	**Train-based Interval**
U-HMM	1.843	[1.508, 2.075]
U-HSMM	1.784	[1.493, 1.998]
M-HMM	1.732	[1.562, 1.984]
M-HSMM	1.691	[1.426, 1.821]
HM-HMM	1.662	[1.502, 1.825]
HM-HSMM	1.593	[1.447, 1.709]

Experiment 2. In the second experiment, the *noisy test data* is used to compare the performance of HMM variations in detecting anomalies. As shown in Table 3.2, H-HSMM outperforms all other methods in terms of precision, recall, and F-measure, and U-HMM has the worst performance. For the same reasons mentioned in Experiment 1, HM-HSMM has the superior performance. Using a hierarchical HMM results in a higher recall where the anomaly score of each data point is verified by referring to higher level models and the final decision is made based on more contextual knowledge. While in security-related data mining applications the prior goal is generally achieving high recall, HM-HSMM gains higher precision without losing recall.

3.6.3 Baseline methods

In this part of experiment, AnomalyTracker is compared with the following baseline methods:

ARIMA-based anomaly detection. ARIMA [51] is a linear time series forecasting model that works based on the linear dependency of the future values on the previous data.

Table 3.2: Performance of variations of HMM using noisy test data

Model	Precision	Recall	F-measure
U-HMM	70.61	71.6	71.10
U-HSMM	76.06	75.6	75.83
M-HMM	71.79	72.8	72.29
M-HSMM	75.28	79.2	77.19
HM-HMM	76.99	79	77.98
HM-HSMM	78.87	81.4	80.12

SARIMA-based anomaly detection. SARIMA is a variation of ARIMA considering the seasonality aspects [39].

Local Outlier Factor (LOF). LOF is a density-based method that compares the local density of a point and its neighbors to find the isolation degree of the point [37].

Log-likelihood-based HSMM. This method assigns an anomaly score to test windows based on the difference of log-likelihood of their state-sequence with a normal interval [141].

Mismatch-based HSMM. This method counts the number of mismatches and assigns anomaly score to test windows based on the number of improbable emission and transitions happened in their predicted state sequence [281].

For running the experiment, *anomalous samples* are used. As presented in Table 3.3, AnomalyTracker outperforms all other baseline methods significantly. AnomalyTracker outperforms the results from the overall second best method by 19%, 8% and 14% for the precision, recall, and F-measure, respectively.

Table 3.3: Performance of baseline methods

Method	Precision	Recall	F-measure
LOF	67.09	79.76	72.86
ARIMA	62.66	68.03	65.24
SARIMA	65.21	76.3	70.32
Log-likelihood-based HSMM	63.78	84.37	72.65
Mismatch-based HSMM	55.75	65.76	60.34
AnomalyTracker	82.36	92.13	86.97

The stronger performance of AnomalyTracker has two important reasons: *1)* Hierarchical modeling provides AnomalyTracker the possibility of following the trend of flow level in various contexts using relevant customized models. *2)* The regression-based module provides it with more knowledge of trend of data which addresses and covers important weaknesses of HSMM-based models. Using this module, AnomalyTracker is able to track the actual trend of data in cases that HSMM is not able to do so. The experimental results show that combination of Hierarchical HSMM and regression can follow the trend of data in any special contexts very well in comparison with other methods. As discussed in Section 3.4.2, an important advantage of AnomalyTracker is detecting the exact point of

anomalies rather than only differentiating between anomalous and normal data sequences, which is not the case in the studied baseline methods.

Log-likelihood-based HSMM outperforms mismatch-based HSMM, meaning that just counting the number of mismatches [281] is not efficient in this problem domain and log-likelihood-based HSMM is a more reliable approach. Comparing SARIMA and ARIMA, we again notice the importance of handling the seasonality issue as SARIMA outperforms ARIMA. The result of LOF shows that this method is not successful in our problem domain since it only relies on the statistical divergence of data and not able to handle seasonality and trend of data.

AnomalyTracker performance for detecting different anomaly types is presented in Table 3.4. AnomalyTracker is able to detect all maximum flow and minimum flow anomalies while it has weaker but still good performance—with the recall of 89% and 85%—for more complex anomaly types, continuous overflow and frequent overflow. Detecting these types of anomalies is more challenging since their behavior is very similar to normal sequences in the context. A successful anomaly detection method for detecting these complicated anomalies should be very sensitive to small changes which can result in increasing false positive ratio and consequently decreases the precision value. While in security-related data mining application the focus is on increasing the recall value to detect positive sample as much as possible, AnomalyTracker is able to present high precision value comparable to its recall value which is a significant advantage of this method.

Table 3.4: Performance of AnomalyTracker for different anomaly categories

Anomaly Type	**Precision**	**Recall**	**F-measure**
Maximum Flow	100	100	100
Minimum Flow	100	100	100
Continuous Overflow	81.49	89.1	85.13
Frequent Overflow	77.91	84.76	81.19

3.7 Conclusions

This chapter proposed AnomalyTracker, a method which traces anomalies in SCADA-based systems using an improved variation of hierarchical HSMM. AnomalyTracker is able to detect various types of anomalies in the context of critical infrastructure protection. Considering the trend of transitions between data points boosts the AnomalyTracker performance to significantly outperform the baseline methods in detecting collective anomalies. An important advantage of AnomalyTracker compared to the existing methods in the literature is to employ a multi-granular approach for verifying the anomaly score of the test windows under investigation. Furthermore, switching to the regression-based mode in cases where HSMMs are not able to follow the trend of normal data is another novelty of the proposed method. To evaluate AnomalyTracker, real-world data from the municipal water supply system is used and extensive experiments is performed in which AnomalyTracker outperforms

the best baseline method by 19%, 8% and 14% for the precision, recall, and F-measure metrics, respectively.

Cybersecurity research is a principal area of innovation to support our economy and the security of societies worldwide. Building a situational awareness platform that addresses the various elements of physical and cyber threats to critical infrastructure is crucial and cannot be overlooked. We believe the proposed situation analysis framework (CSAF) and the anomaly detection methodical approach (AnomalyTracker) can open new perspectives in cyber-physical security of common and widely deployed critical infrastructure.

Chapter 4

A Novel Behavior Modeling Algorithm

Nowadays, automation is essential for operating equipment and monitoring conditions of machinery, production processes and plants; it enhances the efficiency and quality of service delivery, the safe operation and also the protection of critical assets in case of internal or external disruptions, for instance, caused by system failures, physical attacks or cyberattacks [10, 152, 292, 300].

Situational awareness requires continuous situation analysis on massive volumes of time series data from heterogeneous sensor networks. The system behavior being observed is usually stochastic, making the next value prediction, or Time Series Forecasting (TSF), a tricky and challenging task that often needs customized methods. Although the general TSF problem has a rich history within the data mining and statistical machine learning community [35, 205], and a variety of stand-alone as well as ensemble methods have been proposed to handle the problem [55, 76], it is still one of the pivotal topics in the context of time series analysis.

Embedded control systems for operating critical infrastructure interact with their *physical environment* by reading data signals and generating control signals in response. Hence, latent stochastic, or semi-stochastic, external factors influence the system behavior and complicate the prediction problem. Devising a TSF method that has high average accuracy as well as small error deviation is the main goal of this chapter. Identifying hidden patterns and selecting an appropriate model that fits the observed data well and also carries over to unobserved data is not a trivial task. Beyond accuracy related measures, the *quality* of the forecasting method is essential, meaning for each data point there is a reliable estimation of the next value [35, 60, 96]. For example, anomaly detection is an important application of prediction that needs a qualified value estimation; otherwise, measures like MAE (Mean Absolute Error) are not helpful in setting any threshold for differentiating anomalies from normal observations, increasing the risk of high false alarms. The same scenario applies to quality of service applications.

Advanced statistical methods address different characteristics of time series by transforming non-stationary time series into stationary ones, enabling process modeling and prediction [35, 67]. The main drawback of such parametric methods is their restriction to a predefined model, which is an ideal case that does often not reflect reality. System behavior may be affected by external factors that are difficult to capture; for example, the behavior of an overloaded device may deviate so far from

what is considered a normal operation that it becomes impracticable to reason about the effective variables [195]. Further, unknown latent factors can lead to heteroscedasticity and non-Gaussian errors, contradicting the basic premise of parametric methods. Thus, under such circumstances, TSF is difficult for parametric methods, even for very near subsequent steps.

This chapter proposes a novel Multi-Branch Predictor Framework (MBPF) for time series analysis and prediction as an ensemble of deep learning and a parametric method, called TBATS, which stands for Trigonometric Box-Cox transform, ARMA errors, Trend, and Seasonal components [67]. Compared to nonparametric methods, parametric methods have certain advantages for time series prediction, they are more accurate in linear parts and also more robust against overfitting than nonparametric methods.

In the first phase, both TBATS and the deep network are trained individually. To address the problem of possible overfitting and keep a good generalization, the deep network is trained using transformed versions of data based on various seasonality patterns and statistical properties. Then, in the second phase, the results obtained from deep learning and TBATS are exploited by applying a deep neural network (NN) as the backbone for voting between the two methods, taking advantage of its ability to learn discriminatively the mapping between inputs and outputs. Additionally, contextual and structural information about the dataset in the training network are used. This approach takes advantage of extracting features at different time scales to improve accuracy without compromising reliability in comparison with the state-of-the-art methods. The experimental evaluation is performed based on real-world SCADA data from a municipal water management system to show that our proposed method outperforms the baseline methods evaluated here. MBPF beats the other methods regarding accuracy as well as reliability.

4.1 Problem definition

Considering the formulated multivariate stream (X_T), the main goal of this project is designing a *Time series forecasting (TSF)* or *behavior predictor* technique to predict the next observation values X_{T^+} of the representative stochastic time series (where $T^+ = \{t+1, ..., t+l \mid l \geq 1\}$), based on the previously observed time points $\{t-w, ..., t\}$, where w defines the length of the lookback window. A reliable accuracy is desired, meaning symmetrically distributed prediction error, with limited standard deviation.

$$\mathrm{TSF}(x_{t-w,...,t}) = X_{T^+} \tag{4.1}$$

4.2 Background

Time series forecasting has a long history and a variety of parametric and nonparametric techniques such as ARMA, ARIMA, Regression, Functional coefficient model, SVR family, and Neural Networks have been proposed for time series forecasting [205]. Applicability and efficiency of each of these methods depend on the problem domain and properties of the time series data. We briefly recall

some important methods in this field. Box-Jenkins models [35] have been developed by statisticians as systematic methods applying Auto-regressive Integrated Moving Average (ARIMA) to find the best fit of a time series model to previously observed values. These models use an iterative three-stage approach, including model identification and selection, parameter estimation, and model checking steps.

To apply a forecasting model to a non-stationary time series, it needs to be transformed into a stationary one [195]. In this process, a time series data is decomposed into three components of *trend*, *seasonality*, and *random remainder* that can be interpreted as stationary and predictable parts of the data.

The TBATS model is the most generalized version of traditional seasonal investigation models which is able to capture multiple seasonality. Since TBATS can have a very large number of states, it is able to detect a huge number of values for seasonal patterns [67].

Also, there are some other widely-used and well-accepted traditional data-driven techniques like *Support Vector Regression (SVR)* [246], which map time series data to a higher dimensional feature space using a kernel function to learn from non-linear time series. However, they are facing a variety of other challenges, like parameter learning, and limited capacity, which restrict their applicability.

Traditional algorithms are not scalable enough to handle complex behavioral patterns in big data. Besides, reducing dimensions or selecting the most important features out of thousands of dimensions is not a trivial task for traditional models. While neural network models [49, 53, 80, 135, 152] and deep models are qualified to address all of the mentioned challenges.

For instance, Recurrent Neural Networks (RNNs) are effective structures in modeling time series data. The reason is the ability of RNNs in adapting their backpropagation algorithm to unfold the network through time [241]. Also, RNNs keep the memory of the previous inputs by limiting how learned weights change. Since the vanishing gradient is the major drawback of RNNs [26], LSTM introduces a complex block of computing units with specific input gates to trap the errors without losing the sequential properties of input data [98, 132]. Recently, many other TSF deep models have emerged [69, 228], which mentioning all of them is out of the scope of this document and I refer the reader to the following research surveys [6, 164, 204].

4.3 Addressed Challenges

TS modeling and anomaly detection is still a challenging and ongoing topic in machine learning research [287]. Identifying hidden patterns and selecting an appropriate model that fits the observed data well and also carries over to unobserved data is not a trivial task. Beyond accuracy-related measures, the quality and reliability of forecasting is essential for anomaly detection tasks.

Even though both traditional model-based and advanced data-intensive methods have hugely contributed to this field, they suffer some drawbacks. Extremely data-intensive models like deep learning have achieved very high accuracy in TS analysis and handling non-linear complex patterns; but because of not having any confidence interval and being exposed to potential changes in the data,

they may provide biased and off predictions. Whereas, the strong restrictions and assumptions help the traditional statistical methods to control forecasting and provide a realistic confidence interval, especially in presence of linear sections of TS. However, we know that the traditional statistical methods suffer some fundamental limitations in detecting seasonality, trends, and dependencies among data points in complex time series [108, 136, 195]. Thus, this study takes advantage of both approaches and proposes a new hybrid model that is able to handle more complicated TS consisting of both linear and non-linear components.

Also, the lack of sufficient data to represent data distribution and noisy data are some other problems, which cause none of the existing standalone techniques can be considered as an ideal predictor. This chapter proposes an approach that improves accuracy without compromising reliability by taking advantage of the extracted features at different time scales.

4.4 Proposed method

I propose a novel Multi-Branch Predictor Framework (MBPF) to perform time series analysis and prediction. The main pillars of the proposed methodical framework, and the way that it addresses the challenges to enhance time series prediction problem is discussed here. MBPF captures the observed stochastic time series and predicts the next value by introducing two novel contributions: *1)* applying distinct augmentations of data to feed a *deep network*, which leads to higher accuracy and lower deviancy; and *2)* applying a special ensemble of the *deep network* and *parametric* components to increase the accuracy in the potential linear and semi-linear intervals. As illustrated in Figure 4.1, MBPF consists of three consecutive phases: *1)* Pre-learning transforms and prepares the input data by extracting influential contextual factors, removing some outliers, and finding potential complex seasonality patterns; *2)* Individual-learning, which itself contains two main components: *a) multi-branch deep network* and *b) parametric learning*. The former learns particular features of different variations of the input time series and the sequential relation of observations in local branches. Each local branch is considered a feature learning module based on one specific modality of the time series, and the joint representation is finally obtained through multimodality layers. The latter fits the best parametric model on the pre-processed time series; *3)* Consensus obtains concordance through an *ensemble* component by training a Multi-Layer Perceptron (MLP) on the multi-branch deep network and parametric based prediction as well as contextual and structural information. The following elaborates on the rationale behind the building blocks of each phase.

4.4.1 Pre-learning

This section explains the main components of the pre-learning phase.

1. Feature extraction. To extract certain meaningful features, the primary characteristics of the input dataset are to be analyzed and checked as explained below. The purpose is twofold: first, it

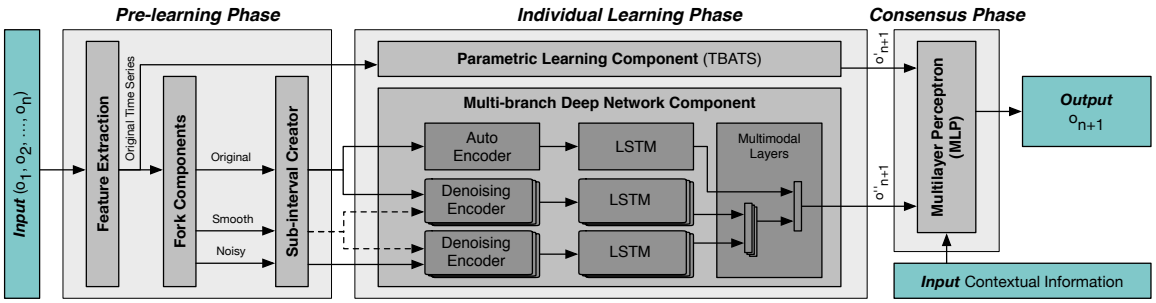


Figure 4.1: Structural architecture of the MBPF framework

leads us toward deciding whether or not the parametric method is sufficient for the prediction task; second, it helps in making required adjustments for the subsequent levels.

• *Seasonality evaluation* analyzes the type of relationship between values observed over time. If the relationship is either nonlinear or inconsistent, the parametric model may perform poorly. To this end, a time series decomposition approach based on an *additive decomposition model* [190] is used. The input time series can be represented as $O_t = T_t + S_t + Z_t \quad (t = 1, 2, \ldots, n)$, where T_t is the *trend* component, S_t is the *seasonal* component, and Z_t is the stationary *random noise* component.

• *Curvilinearity adjustment*. The parametric methods cannot handle the non-periodic cycles, therefore, fading them is preferable. Based on the (Partial) Auto Correlation (PAC) analysis [190], the combination of appropriate transformation functions (e.g., *d-lag difference*) can be determined. The residual of the transformed data is then analyzed to find artificially inflated standard errors. If the *p-value* of the normal residual is lower than a threshold, parametric models are inappropriate for prediction.

• *Outlier elimination*. Another important analysis prior to modeling and prediction is removing solid exceptional phenomena as outliers. Although various methods show different sensitivity and robustness in the presence of outliers, empirical studies show that removing these kinds of definite abnormalities will lead to a more accurate predictor, especially in data-driven methods [35]. In this method, in addition to decreasing the variance of residual in the parametric learning component, it helps to select a more realistic variance for Gaussian distribution which MBPF applies to generate the *noisy sequence*.

• *Contextual feature analysis*. Contextual features (explanatory variables in statistics) should be found and considered in the method. Correlation analyses between the main variable and other explanatory variables is performed by evaluating the following properties: *1) Heteroscedasticity*: A specific parametric model is fitted based on the main variable and the heteroscedasticity test [190] is applied on the model to identify the normality of residuals. If the *p-value* is relatively small, then we reject the heteroscedasticity assumption. *2) Multicollinearity*: Redundancy is a serious issue and such variables should be removed from the model (or possibly be modified by creating interaction variables or increasing the sample size) to avoid having a biased, unstable, and unreliable model. The Variance Inflation Factor (VIF) test [190] is used for this purpose.

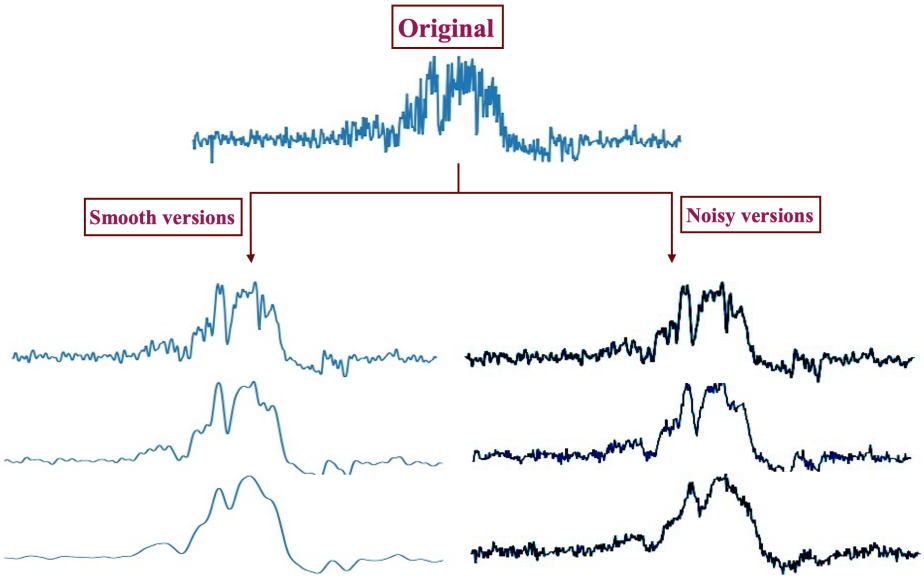


Figure 4.2: An example of fork component data augmentation in three levels of granularity

2. Fork component

Neural deep networks can easily get overfitted with limited accessible data. Therefore, a data augmentation approach should be used toward fading random noises. In real-world time series, an often inescapable phenomenon is random noises and deviations that affect discriminative and specific patterns in the data. [62]. As Figure 4.2 illustrates, the dataset is augmented by preparing three variations of the input sequence to learn primary features:

• *Original sequence*. The original version of the time series is useful to extract features directly.

• *Smooth sequence*. The idea behind augmenting data with a smooth version of the time series is based on an assumption that the distribution of random parts leads to a constant mean over time. Convolution transformation based on the Moving Average Box (MAB) approach is used to create a smooth sequence with lower frequency. However, instead of taking random parameters for the length of MAB, the length of potential levels of seasonality is used. Considering O_t, the convolution-based moving average, transforms O_t into $Smooth_t$, where $MAB^l = \{(1/l, \ldots, 1/l) : l \in \{multi\text{-}seasonality\}\}$.

$$Smooth_t^l = (O * MAB^l)_t = \sum_m o_m mab^l_{(t-m)} \tag{4.2}$$

• *Noisy sequence*. Additive White Gaussian Noise (AWGN) is a basic noise model used in the field of information theory to mimic the effect of many random processes occurring in nature. Therefore, to prevent the network from getting overfitted, we can apply AWGN, which only increases the noisiness of the data but does not affect its overall distribution and pattern. The formula of noisy time series

is as follows, where Int_l is defined as $\max(\mid Smooth_i^l - o_i \mid)$.

$$K_t^l = \{(k_1, \ldots, k_n) \mid k_i = Smooth_i + Z_i \sim \mathcal{N}(Smooth_i, Int_l)\} \tag{4.3}$$

3. Sub-interval creator

Analysis of all available log data in one step is not usually very helpful. This is mainly because the representative features can easily get lost in long sequences; therefore, the input should be broken into smaller intervals. However, window-size is a very fundamental concept in time series analysis (e.g., prediction), and finding the best window-size is often problematic. On the one hand, there is a direct relation between the span of cycles, seasonality, and trend change-points with lookback windows; meaning that a lookback covering all of the change-points provides more knowledge. On the other hand, the appropriate size of the lookback window is highly dependent on the forecasting mechanism of different methods. Sometimes, more information misguides the method and diminishes the role of closer intervals. Moreover, a longer lookback means more neurons, which indeed needs more training data to avoid overfitting. Therefore, the difficulty lies in a tradeoff between the effective score for an observation's significance versus gaining more knowledge regarding change-points over time. The following formula is considered to balance the number of hidden layers, inputs, and the size of the training set. α represents the degree of freedom in the formula and the generalization level [72].

$$N_h = N_s/(\alpha * (N_i + N_o)) \tag{4.4}$$

4.4.2 Individual learning

This phase trains a deep network and a parametric model separately. The main reason for training these models separately is to allow each method to try its best to fit an independent predictor and to provide its best estimations. The final model forecasts the next value based on the results of two main components.

Deep network learning

This component is the heart of MBPF and trains a predictor for the next value. It is responsible for *parallel* local learning of different representations of each time series and also for finalizing the learning process through multi-modality learning. Each branch consists of an autoencoder and a Long Short Term Memory (LSTM) which learns different representations of input time series. As shown in Figure 1.2, we have the following three branches:

1) Original sequence-based branch: The autoencoder of this branch, which is a *compression autoencoder*, is trained based on the same input-output as the original sequence. The learned representation feeds the following LSTM.

2) Smooth sequence-based branch: For each smooth representation of the original data, the same process is followed: First, pre-train a denoising autoencoder, which maps the original sequence on

$Smooth_i^l$. Then, the following LSTM learns the sequential relation based on the data representation that the autoencoder provides. As mentioned in Section 4.4.1, the number of learning sequences in this branch is equal to the number of identified levels of seasonality in the data.

3) Noisy sequence-based branch: It has the same structure as the previous one, except in the pre-training of the denoiser, it maps the *noisy sequences* as input (K_t^l) to the smooth version $Smooth_i^l$. Let's briefly go through the definition and reasoning behind applying the following elements in our framework.

• **Autoencoder.** Empirical research studies [85] show that deep networks with more than a few layers do not outperform simple structures, especially when the training has been initialized by random weights, the result can go even worse. But pre-training techniques [131] overcome these challenges by supporting a better generalization of the training dataset through guiding the learning process towards a lower minimum of the empirical cost function. Autoencoder is an unsupervised pre-training technique which is very powerful in extracting patterns of data [85] based on minimizing different approximation of the log-likelihood.

• **Denoising autoencoder**. Based on [276], "A good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input." This theory uses two important points: 1) targeting the robustness of the higher-level representation, and 2) making the extracted features focus on the fundamental structure instead of the random part. From the mathematical point of view, the noisy parts are farther from real ones and a successful denoising method should be able to substitute low probable points with high probable ones.

• **Long short term memory.** LSTM by applying special neuron structure is a very appropriate Recurrent Neural Network (RNN) to extract the sequential features of time series. Its memory cells have the ability to store information over an arbitrary period of time. As illustrated in Figure 4.3, three gates control the information flow of the memory cell. Each gate gets the same input as the input of the neuron, *forget gate* decides what information should be thrown away, *input gate* decides what new information should be stored in the cell, and finally *output gate* determines the output of the cell.

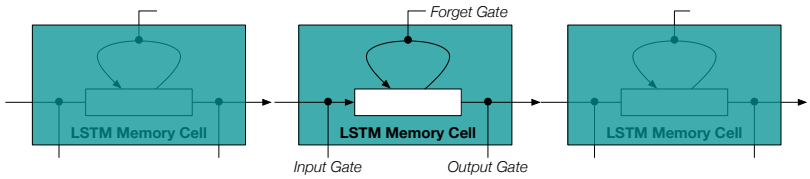


Figure 4.3: Structure of each LSTM neuron

• **Multimodal layers.** After parallel training each branch, the obtained features should be combined. The first layer combines the results of the noisy and smooth models and the second layer concatenates the results of the parallel components and provides the network with the possibility of multi modalities prediction, a state-of-the-art deep learning method [211].

Parametric learning

This component prepares the final result of the parametric model, which has been configured based on extracted statistical properties of the input time series. As aforementioned, the TBATS is applied which has proved its efficiency in comparison with other parametric methods in coming up with complex seasonal patterns. Going through the details of this method is out of the scope of this study. Relying on a new method that has greatly reduced the computational burden of maximum likelihood estimation, using exponential smoothing and trigonometric based decomposition are the key features of this time series. After fitting the best TBATS model on time series, its estimation for training data will be collected for the next phase.

4.4.3 Consensus phase

This phase concatenates the results of the parametric model in the presence of *auxiliary information* with the results of the deep network and gets the final output through a Multi-Layer Perceptron (MLP). Auxiliary information is considered as related contextual dimensions and time-related properties of the next expectation. The reason for applying these structural and contextual properties as an input to the second deep network (MLP) is to generate the best prediction based on the results obtained from both components with more information. This is another contribution of the proposed framework, distinguishing it from existing ensemble-based predictors. Instead of simply voting between the parametric and nonparametric components, a second deep network learn how to select between two results. Therefore, it can potentially learn any arbitrary relationship between inputs and outputs.

4.5 Experimental Evaluation

This section presents the experimental evaluation of MBPF in comparison with the baseline methods.

4.5.1 Experimental setup

For the experimental evaluation, SCADA data from the public water supply system of the City of Surrey in B.C., Canada (total population > 500 thousand) for the time period of 2011–2014 is used. The dataset contains historical data of various water stations including drinking water, drainage, sanitary and vacuum stations. For each water station, the SCADA data is a multivariate time series with one data point per minute comprised of several features including *Inflow*, *Flow-Level*, *Air-Temperature* and *Air-Humidity*.

Seasonality patterns indicate fluctuations in residential drinking water consumption, providing an important data characteristic. Seasonality patterns at various aggregation levels, which can contribute to detecting irregular patterns, are identified. We can observe daily, weekly, monthly and seasonally strong patterns in the water consumption. Not surprisingly, *Air-Temperature* is the main factor in shaping seasonality patterns.

We break down the prediction task into short-term and long-term forecasts. Specifically we differentiate here between *hourly forecasts* and *daily forecasts* to predict the average water *Flow-Level* for the next hour and the next day. Short-term prediction tends to be more relevant to safety and security objectives such as detecting cyberattacks, whereas long-term prediction is more geared towards improving water management and quality of service by predicting future trends based on regular patterns in historic data.

As explained in Section 4.4, the noisy and smooth versions of data are generated based on the detected levels of periodicity. For example, in the process of generating smooth version of time series for daily forecast, we use convolution transformation based on three sizes of MAB including: 7, 31, 90 for the weekly, monthly and seasonally periodicity, respectively. In accordance with each of the smooth versions, another noisy version is generated by applying additive white Gaussian noise. The variance of added noise is set based on the maximum difference of data points in the original version compared to the smooth ones.

To compare the accuracy and reliability of the proposed method with the studied baseline methods we use four different measures. We use two commonly adopted scale-dependent measures Mean Absolute Error (MAE) and Root Mean Square (RMSE) working based on the absolute errors and squared errors:

$$MAE(O_t^{'}, O_t) = \frac{1}{N}\sum_{i=1}^{N} |o_i - o'_i| \tag{4.5}$$

$$RMSE(O_t^{'}, O_t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (o_i - o'_i)^2} \tag{4.6}$$

where o_i is the predicted value and o'_i is the observed value. The RMSE measure is minimized by the conditional mean, while the MAE is minimized by the conditional median. Using these two similar measures helps us to recognize whether the learning method is fitted in an unbiased way. Moreover, since RMSE exaggerates error values, it is able to measure the reliability of methods by considering the limited deviations between the forecast and original values.

The third measure we use for evaluation is Absolute Deviation (AbsDev). AbsDev measures the reliability and quality of forecasting method based on the deviation of residuals; smaller values of AbsDev corresponds to a more symmetric error distribution:

$$AbsDev(O_t^{'}, O_t) = \frac{1}{\sum_{i=1}^{N} o_i}\sum_{i=1}^{N} (|o_i - o'_i|)^2 \tag{4.7}$$

Relative Errors (ReErr) is another measure to evaluate the efficiency of forecast methods in comparison with the random walk model. For generating the next value of a time series data, a random walk model adds a random error value with zero mean to the current value of the time series.

$$ReErr(O_t^{'}, O_t) = \sqrt{\frac{\sum_{i=1}^{N}(o_i - o'_i)^2}{\sum_{i=1}^{N}(o_i - o_{i-1})^2}} \tag{4.8}$$

4.5.2 Evaluated methods

We use three sets of baseline methods in comparison with MBPF. The first set consists of long-established methods.

Auto Regressive Integrated Moving Average (ARIMA) is one of the general classes of linear equations to fit on "stationary" time series for better estimating and forecasting future data. *TBATS* is a variation of ARIMA which can handle complex seasonal patterns [67]. *Support Vector Regression (SVR)* is an extension of SVM for regression based on ϵ–non–sensitive loss functions. This model computes a linear regression function in a high dimensional feature space mapping the input data by means of a nonlinear function [246]. *Multilayer Perceptron (MLP)* is one of the feed-forward neural networks that maps sets of input data to a set of respective outputs. MLP has been used widely in time series prediction [160].

The second set of baseline methods consists of RNN-based methods that memorize temporal dependency of data points in time series to predict the future. *Stacked LSTM* is a stack of two or more LSTM layers which allows for greater model complexity similar to multi-layer networks [105, 146]. *Regularized LSTM* is an LSTM with dropout layers to decrease the chance of overfitting. The dropout operator corrupts the information carried by the units, pushing them toward performing their intermediate computations more robustly [293].

Regarding the last set of evaluated methods, we perform an ablation study based on different variations of MBPF to understand the effect of MBPF components to its performance. The *individual learning phase* has two major components, *parametric learning* and a *multi-branch deep network*, which are unified through a *consensus phase*. *MBPF-1BDN* is equal to one autoencoder and one LSTM. It uses only the *original sequence based branch* (see Section 4.4.2 for details) of the *multi-branch deep network* component, while ignoring other branches. This is the simplest variation of MBPF and is comparable to the method proposed in [104]. *MBPF-2BDN* uses two branches (*original* and *smooth*) of the *multi-branch deep network* component, while ignoring the *noisy* branch. *MBPF-MBDN* uses all three branches of the *multi-branch deep network* component.

It is noteworthy that the *MBPF-1DBN*, *MBPF-2DBN*, and *MBPF-MBDN* configurations all ignore the *parametric learning* component and the *consensus* phase. All of the methods are trained using a training set and are applied on a test set to predict the next observation. Therefore, all of the methods that use a lookback window in their prediction process may take advantage of the available data till o_i to predict o_{i+1}.

4.5.3 Experimental results

This section compares the performance of MBPF with the evaluated baseline methods and explains how the elements of MBPF contribute to the performance.

Comparison with baseline methods. Tables 4.1 and 4.2 show the best obtained results from different configurations of baseline methods and MBPF with respect to the reference measures. These results show that MBPF outperforms all other baseline methods in terms of both accuracy and reliability. MBPF surpasses them by 10%, 11%, 16% for the measures MAE, RMSE, and AbsDev, respectively. Further, the ReErr value of MBPF is about 0.2, which is significantly better than the best baseline method with ReErr of 0.5.

This experiment supports the inefficiency of parametric and nonparametric methods used separately for TSF. The main drawback of *ARIMA* and *TBATS* is using strict assumptions in the data fitting process, as our experiments indicate. Moreover, being sensitive to small changes in the training data can cause overfitting and performance decreases in the test phase. This issue is observed in some of the evaluated methods like *Stacked LSTM*.

The MAE measure of MBPF is improved by 14% and 6% for hourly and daily forecast, respectively, in comparison with the second-best baseline method. Achieving the lowest MAE value for MBPF in both training and test data shows that the proposed method using a multi-branch deep network is able to efficiently learn the specific features of time series without getting overfitted to random parts of data. Moreover, MBPF has the lowest RMSE compared to the baseline methods. This measure for MBPF is improved by 15% and 7% for hourly and daily forecast, respectively, compared to the second-best baseline method. The results of MBPF obtained for daily and hourly forecasts are dissimilar due to the different size of the training samples in these two settings. While MBPF outperforms the baseline methods in both settings, it can achieve a better performance for hourly forecast.

In critical infrastructure related applications the *reliability* of forecasting is a vital issue. Reliability can be measured in terms of the deviation of forecast errors. So that if the forecast error follows the normal distribution, then the predicted values are more reliable. Achieving the best AbsDev shows that MBPF is a more reliable approach without significant skewness in forecast values. Having close values for RMSE and MAE leads to the conclusion that the error values of different data points are close as well. This result strongly indicates reliability of MBPF in the forecasting process. As discussed, the close values of of MAE and RMSE for a forecasting model, $MAE \cong RMSE$, show that such a model is unbiased, another indication of quality of prediction.

MBPF works two times better than the second best baseline method with respect to relative errors, which is a significant result. This result implies that MBPF is able to learn deep patterns of time series data to forecast the next value. Comparing different variations of MBPF, as Figure 4.4 shows, MBPF-*2BDN* outperforms MBPF-*1BDN* on the training data but not on the test data, which can be a sign of model overfitting. On the other hand, MBPF-*MBDN* is able to significantly outperform both MBPF-*2BDN* and MBPF-*1BDN* methods on the training and test data. This comparison highlights

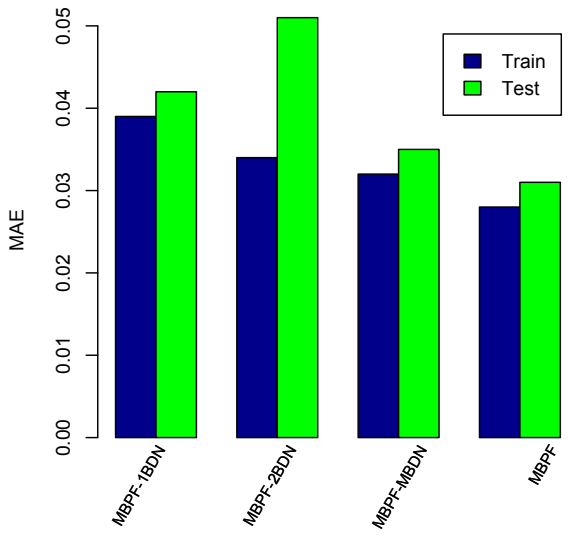


Figure 4.4: Comparison of train and test on MAE

two important points about MBPF: 1) By taking advantage of smooth and noisy versions of data, the multi-branch augmentation approach in MBPF is able to capture and memorize the main underlying structure of time series; 2) Obtaining very close values of performance measures on the test and training data shows that the augmentation approach is not overfitted to random parts of the observed data. Adding a noisy version of data empowers MBPF to distinguish fundamental features of data from random noises.

Lookback window size effect. The time series window size, or lookback window, is an important parameter affecting a forecasting method's performance. The lookback window size decides the number of previous observations that will be used in forecasting the next value. Based on the given relation in Section 4.4.1 by keeping the value of α between 4 to 8, different sizes of window, including 20, 50, 100, 200, 500, and 1000, are tried to find the best lookback size for MBPF. Our experimental results show that MBPF works best with the lookback windows of size 200 for hourly and 100 for daily forecasts. Even though the size of lookback window is an influential factor in the MBPF performance, changing window size has a similar effect on values of the evaluation measures. For instance, the performance of hourly forecasts with a lookback window size of 200 is improved by 15%, 13%, 9%, and 26% for *MAE, RMSE, AbsDev* and *ReErr* measures, compared to the second best results with the lookback window size of 500.

Autoencoder comparison. The performance of different autoencoders using *loss value* and *RMSE* measures are evaluated. Loss value measure computes the deviations of input data from the fitted values. This measure evaluates the ability of autoencoder in preserving and reconstructing the input data. The output of each encoder is used as the input of an LSTM network. Then, RMSE measure is used to evaluate the prediction accuracy of each network to verify the effect of using autoencoder on the performance of LSTM network. Smaller values of *RMSE* show higher quality of

Table 4.1: MBPF variations and the evaluated baseline methods performance in hourly forecast

Method	**MAE**	**RMSE**	**AbsDev**	**ReErr**
ARIMA	9.617	9.619	30.84	10.991
TBATS	0.107	0.142	0.345	2.069
SVR	0.042	0.056	0.137	0.81
MLP	0.036	0.049	0.115	0.713
Stacked LSTM	0.054	0.077	0.142	0.895
Regularized LSTM	0.037	0.048	0.117	0.701
MBPF-1BDN	0.034	0.047	0.106	0.628
MBPF-2BDN	0.038	0.052	0.099	0.647
MBPF-MBDN	0.033	0.046	0.085	0.286
MBPF	**0.029**	**0.040**	**0.079**	**0.193**

Table 4.2: MBPF variations and the evaluated baseline methods performance in daily forecast

Method	**MAE**	**RMSE**	**AbsDev**	**ReErr**
ARIMA	9.639	9.639	33.19	176.827
TBATS	0.166	0.184	0.572	3.368
SVR	0.059	0.071	0.147	0.939
MLP	0.057	0.086	0.149	1.018
Stacked LSTM	0.058	0.078	0.150	0.988
Regularized LSTM	0.033	0.046	0.0901	0.45
MBPF-1BDN	0.042	0.055	0.133	0.797
MBPF-2BDN	0.051	0.075	0.141	0.802
MBPF-MBDN	0.035	0.047	0.103	0.282
MBPF	**0.031**	**0.043**	**0.084**	**0.251**

an autoencoder in obtaining an appropriate generalization of input data without losing its underlying structure.

While in many vision applications convolutional autoencoders could outperform other types of autoencoders, in time series analysis context *stacked autoencoder* outperforms all others, as shown in Table 4.3. This result indicates that in our studied domain a convolutional autoencoder is not able to memorize the long-range dependency among data points and consequently the structure of time series. Moreover, based on this result we can conclude that a 1-layer autoencoder is not strong enough to learn the complex and nonlinear aspects of time series data. The result of applying the *stacked autoencoder* on one sequence of test data has been illustrated in Figure 4.7. The input sequence shown in red color is encoded into another sequence in more sparse representation, displayed in green color; then the encoded version is perfectly decoded to a smooth representation, shown in blue

Table 4.3: Comparison of denoising autoencoders

Concretization	Train-Loss	Validation-Loss	Train-RMSE	Validation-RMSE
1-layer autoencoder	$4.8285e^{-04}$	0.0011	0.06	0.09
Stacked autoencoder	$7.1088e^{-05}$	$5.8526e^{-04}$	**0.04**	**0.05**
LSTM autoencoder	$2.1245e^{-05}$	$7.5158e^{-05}$	0.06	0.06
Convolutional AE	0.0132	0.0029	0.11	.1

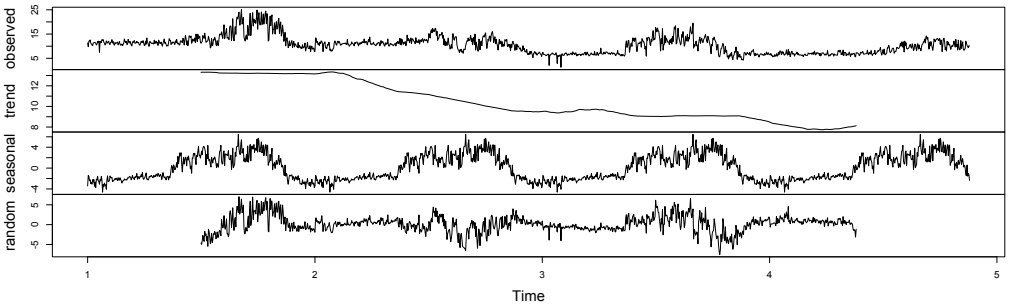


Figure 4.5: Time series and its decomposition components

color. The matching between the inputs and outputs indicates that the stacked autoencoder is able to learn an suitable smoothing function for denoising the time series data.

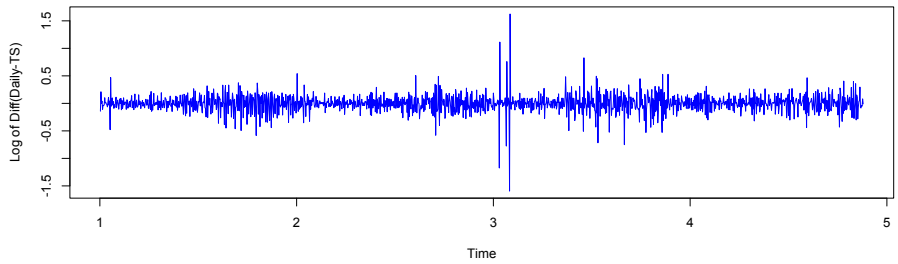
4.5.4 Feature extraction findings

This section briefly discusses important findings of MBPF's feature extraction step.

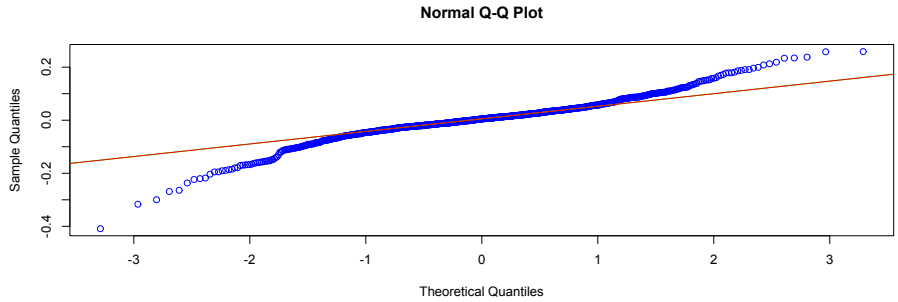
As shown in Figure 4.5, the daily-recorded observations of the studied time series data has a trend and seasonality. Moreover, the seasonality of this data remains after decomposition, creating a skewed error distribution. Because of this property, learning patterns from this time series data needs *curvilinearity adjustment* using an appropriate transformation. This process is generally performed using a hybrid of befitting lag difference and logarithmic transformation of data.

As Figure 4.6a shows, the transformation removes a significant part of non-stationary properties. However, the normality test shows that the error values do not follow a normal distribution. In our case, the result of normality test is 0.001, much smaller than 0.5, the threshold for a normal distribution. This claim is illustrated in Figure 4.6b, which shows the quantile-quantile normal plot of residuals. This finding and the results presented in Tables 4.1 and 4.2 both confirm the inefficiency of parametric methods in modeling the studied time series, especially ARIMA.

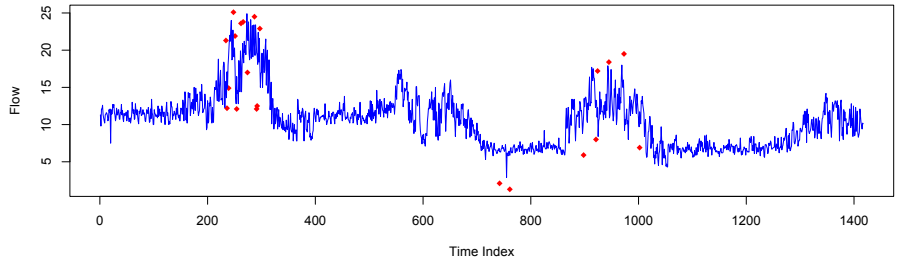
MBPF applies a local-distribution based outlier-detection method to identify the most probable outliers. As Figure 4.6c shows, MBPF is able to substitute the outliers (shown by the red points) with coincident points in the smooth version of the data. Although removing outliers does not lead to a normal distribution of error values, it can improve the variance of the normality test.



(a)



(b)



(c)

Figure 4.6: (a) Transformed version of a time series; (b) Quantile-quantile normal plot of transformed version of time series; (c) Refined time series after removing outliers

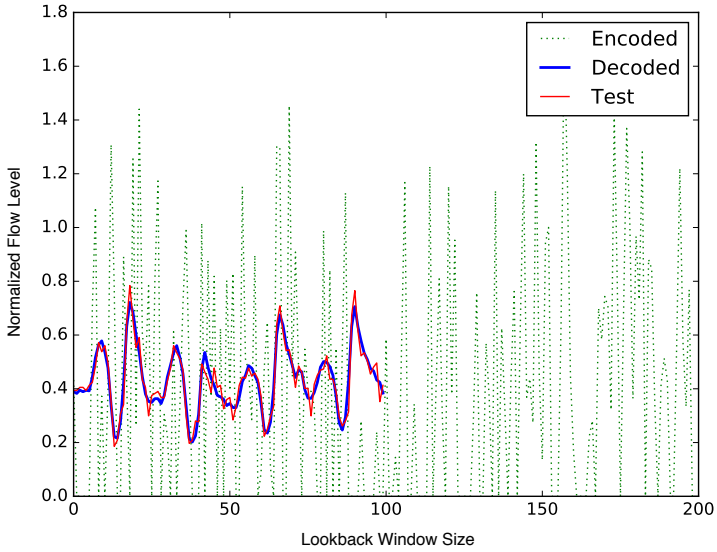


Figure 4.7: Representation of denoising autoenccoder on one sample of daily test data

Table 4.4: VIF of water supply system dataset

Variables	**VIF**
Flow-Level	1.28
Air-Temperature	1.36
Air-Humidity	1.22

In the process of *contextual feature analysis*, MBPF applies the Durbin-Watson test [190] to find autocorrelation in the forecasting errors of the regression analysis. The experiment shows that there is more *heteroskedasticity* in the analysis of water *flow level* without involving *air humidity* or *air temperature* features. Based on the results of the Variance Inflation Factor (VIF) test, presented in Table 4.4, the problem of *multicollinearity* does not occur between the abovementioned features, because their VIF values are about one, a very small value to indicate any significant collinearity. VIFs between 5 to 10 are signs of serious multicollinearity.

4.6 Conclusions

This chapter proposes a novel deep learning based framework, MBPF, for analyzing the structure of a time series and predicting the next value. To address the respective drawbacks when using either parametric methods or nonparametric methods exclusively, the framework takes advantage of ensemble learning based on *TBATS* and *deep networks*. MBPF evaluation through extensive experiments on real-world data from the public water supply system of the City of Surrey, B.C.,

Canada. MBPF outperforms the best baseline methods by 10% for MAE and 16% for RMSE, confirming the efficiency of the framework. Higher accuracy and reliability in predicting the next observation makes MBPF more reliable.

Next observations prediction is a principal area of innovation with many potential applications. Critical infrastructure protection—increasingly vital for the safety of our society and for our economy in dealing with a rapidly evolving threat landscape—is only one example. Beyond prediction, the main aim behind this research, the method can potentially lead to a situational awareness platform that addresses various elements of physical and cyber threats. We believe the proposed framework and time series analysis approach provides new perspectives in analyzing the behavior of common and widely deployed critical infrastructure.

Chapter 5

Dynamic Attack Scoring Using Distributed Local Detectors

This chapter targets the challenging problem of anomaly scoring in distributed systems to filter out irrelevant anomalies and raise alert for the potential attacks. Cyber-physical systems (CPS) [114] integrate embedded computers, software, and networks in control loops routinely used in industrial automation to monitor and control physical processes [74]. The work presented here focuses on *supervisory control* [254], [9] of critical infrastructure and services, i.e, the overall operation of a complex physical system with multiple lower-level control functions and peripheral process controllers that interact directly with the physical system through actuators and sensors. This generally includes common control architectures like SCADA and distributed control systems (DCSs) [157]. For stream signal processing we use a combination of advanced time series analysis and forecasting methods [223] to dynamically detect and analyze abnormal system behavior in near real time.

Contrary to the benefits, in the evolving cyber threat landscape, increasing reliance on automation also increases the attack surface for advanced persistent threats and amplifies the risk of cascading effects. In light of the cyber threats and existing vulnerabilities that expose critical infrastructure to a variety of adversarial scenarios, this work promotes anomaly detection based intrusion detection methods used for cyber situational awareness in the analysis of automated control processes.

Intrusions and security breaches are a reality that can compromise the protection of sensitive data and information, exposing personal identities, intellectual property or financial assets, causing damage that may ruin lives and businesses. Things can still get a lot worse though when attacks on critical infrastructure cripple vital system components physically, e.g., by disabling safety instrumented systems [43]. Beyond transient disruptions, the potential damage could be long-term, threatening human safety beyond comprehension. This situation calls for advanced intrusion detection solutions.

Intrusion Detection. Intrusion detection systems (IDSs) are devices or software applications that monitor networks or systems for malicious activity or policy violations [19] and are essential for protecting industrial control systems and critical infrastructure [157]. Generally, protective measures are either *reactive* or *proactive*. The former deal with emergency response and digital forensics to

mitigate the impact of a security breach and identify the source of the attack; the latter enhance cyber situational awareness, e.g., by analyzing logs to detect suspicious activities pointing to an attack in progress [51, 300]. The rational behind proactive measures, which are the main focus of our study, is mitigating risks—the *probability* of cyberattacks occurring multiplied by the potential *damages* that would result—by improving defense mechanisms. A first step towards enhanced risk awareness and management is developing an understanding of the attack space and the potential system failures [247]. This calls for systematic threat analysis to identify and assess adverse effects on the physical system and its subsystems.

Analytic Framework. Considering the challenges mentioned above, we propose AttackTracker for tracing potential attacks using a hierarchical network of *attack detectors* operating across a distributed control architecture. In this framework, a local detector learns the latent patterns of the local control process it monitors using a Behavior Predictor to find attacks by utilizing a dynamic anomaly scoring scheme. Detectors in higher levels aggregate knowledge obtained from lower level detectors and communicate the system status and severity of any attacks to the system operator. AttackTracker advances the state of the art in online cyberattack detection by making the following technical contributions:

1. Applying a dynamic scoring method which does not only enhance the true positive (TP) ratio but, at the same time, also reduces the false positive (FP) ratio.

2. Improving attack detection through a distributed detector network based on local Behavior Predictors powered by a multivariate temporal convolutional network (TCN) model.

Our experiments and evaluations show AttackTracker can detect cyber threats in a challenging real-world dataset [109] and outperforms state-of-the-art methods. Since the framework proposed here is not specifically customized for this testbed, it can also be applied to analyze any given input sequence of events in other contexts such as network intrusion detection and anomaly detection in trajectories to determine complex *collective anomalies* [51]. This chapter extends our work published as a conference paper [298] by describing and discussing the data exploration, the cyberattack detection framework, and the experimental analysis and results in significantly more detail and depth.

5.1 Problem definition

A supervisory control system as considered here has multiple local control components (each associated with some subsystem). The history of system behavior monitored over some time interval T, comprising t consecutive time steps, forms a multivariate time series X_T. Intuitively, X_T is collectively generated by S local control processes, $S \geq 2$, one for each control component. For a given time series $X_T = \{x_{(1)}, x_{(2)}, ..., x_{(t)}\}$, each element at time (i) is a multivariate data point $x_{(i)} \in \mathbb{R}^S$ composed of S elements, $x_{(i)} = (x^1_{(i)}, x^2_{(i)}, ..., x^S_{(i)})$. Each local control process s, for $s = 1, ..., S$, is a multivariate process with K_s observed features, meaning $x^s_{(i)} = \{f^1_{(i)}, f^2_{(i)}, ..., f^{K_s}_{(i)}\}$, where the number of features may vary from one component to another.

An online ADF is supposed to spot any suspicious anomalous event in near real time by continuously analyzing the stream data. The goal is to assign a score to each data point in the stream data for the likelihood of being associated with an attack. Given the online setting, we do not assume an upper bound for attack scores but allow for any degree of anomaly in future observations.

$$ADF(x_{t+1}|X_T) \rightarrow [0, \infty) \tag{5.1}$$

5.2 Background

Intrusion detection systems (IDS) [19] are vital for protecting industrial control systems and critical infrastructure [157]. Protective measures can be reactive or proactive. The former deal with emergency response and digital forensics to mitigate the impact of a security breach; the latter enhance cyber situational awareness, e.g. by analyzing logs to detect suspicious activities pointing to an attack [51,300]. Proactive measures, which are the main focus of our study, aim at mitigating risks—the *probability* of cyberattacks occurring multiplied by the potential *damages* that would result—by improving defense mechanisms. A first step towards enhanced risk awareness and management is developing an understanding of the attack space and the potential system failures [247].

Intrusion detection. Network security and protection has a very rich and well-defined background and has been addressed in extensive studies [19,100,204,248]. There are three major detection methods applied by IDSs. Signature-based (a.k.a. *misuse detection*) IDS mainly focuses on creating and tracking the signature of previously detected attacks to capture them if they reoccur [248, 252]. An AD-based IDS, the main target of this study, aims to extract the normal traffic patterns so as to identify suspicious abnormalities as attacks [198, 204, 290]; as aforementioned, various approaches such as statistical, knowledge-based reasoning and ML are in principle applicable to find anomalies. The third group is specification-based techniques or hybrid systems, which take advantage of both signature and anomaly detection methods [100, 204]. Lately, the attacks against cyber infrastructure are growing so fast that the volume of new signatures exceeds the ability of human operators to add those signatures to traditional IDSs. Therefore, AD-based IDSs are receiving increasing attention. Further, patterns in the complex stochastic processes are too complicated to be defined by rule-based or traditional ML methods; while studies on applying advanced ML algorithms in new approaches for addressing cybersecurity problems, including detection of zero-day attacks or new variants of malware, have demonstrated significant improvements. On the other hand, the presumption of labeled datasets is too naive and notably, the presumption that features are correlated is too strict in high-dimensional datasets obtained from dynamic real-world systems with hidden ongoing attacks. Thus, ML semi-supervised, profiling based methods are increasingly becoming desired in the stream analysis and threat detection context.

Intrusion detection in cyber-physical systems. Most of the existing research focus on applying ML to perform computer network intrusion detection, while there are not enough studies related to CPS as a new emerging and growing target area for ML-based security algorithms [28]. Despite the

similarities, CPSs bring their own unique threat spaces, because of the discrepancies in their basis and application. Since CPSs and simple embedded controllers for operating critical infrastructure interact with processes in their physical environment by reading data signals and generating control signals in response, there are plenty of latent stochastic, or semi-stochastic, external factors or behavior instabilities that can influence the control process and complicate identifying hidden patterns [301]. Further, anomalies can originate from various sources like cyber or physical attacks, errors in control or communication networks, and other malfunctions with system-wide or local effects. Differentiating *anomalies of interest*—suspicious anomalous behavior indicating a potential security threat—from the set of those anomalies considered irrelevant to security, turns out to be an even more intricate task than finding anomalies [260]. Plenty of the existing methods are inappropriate to be applied on CPS, because they cannot handle correlated features in high dimensional and sparse spaces, differentiate noise from the original behavior and, last but not least, are not applicable on stream data [301].

Nowadays, the remarkable progress of deep learning methods in various contexts such as image analysis and vision [183, 240], object detection [127], IoT and stream analytics [204], as well as the advances in anomaly and intrusion detection [198, 290] have distinguished these types of techniques as capable candidates for extracting hidden patterns in complex data and consequently demystify potential anomalies [28]. Since an IDS should monitor stream data, we need to apply models that are able to analyze sequential and temporal data. For the past few years, long short term memory (LSTM), or generally recursive neural networks (RNNs), were known as one of the best fits for such data. But training them is very challenging because their optimization process suffers from two major inherent problems. First, they are almost unable to recall a very long history, which is highly desired in any historical data analysis context. Moreover, because of the sequential processing of inputs, RNNs cannot take full advantage of parallelization techniques like convolutional neural network models (CNNs). Therefore, temporal convolutional networks (TCNs) [21] are proposed to address these challenges by being faster, highly memory efficient and flexible in adjusting the historical window to the longer input sequences. Accordingly, several AD methods have been proposed based on such powerful deep learning models [164] and some have boosted the accuracy of profile-based AD systems [204].

All in all, none of the existing techniques are off-the-shelf [270, 300] and fully transferable to a new context [12, 270] due to being highly dependent on the system properties and unique vulnerabilities. Thus, they might face a very high false alarm rate or low recall in the new context.

5.3 Addressed Challenges

There are a few challenges toward applying an anomaly detection algorithm to detect intrusions, which are tried to be addressed in this study.

First and foremost, the distributed nature of control systems and their dependence on multiple latent stochastic factors may cause behavior instabilities which complicate customizing a model

to identify hidden patterns and also carry over to unobserved data to trace potential attacks in a complex system. Behavior modeling applied in the training of anomaly detection algorithms has caught more attention in comparison with the decision-making process of post pruning, threshold setting, anomaly scoring, and labeling. Just relying on predictive models is very risky due to uncertain boundaries, continuously evolving behavior, and potential data drifts [260, 284]. Especially, in the real world, usually, AD systems might need to handle heterogeneous set of models based on different detectors. This makes the problem more complicated because a very slight mistuned threshold can unexpectedly lead to an unemployable system that produces a huge false alarm rate or misses most of the attacks by labeling them as normal [38]. And, 'chasing ghosts' confuses data analysts and is a waste of valuable resources after all.

Also, most research efforts are focused on applying machine learning (ML) to perform network intrusion detection, while CPS related attacks, although an emerging and growing area, was almost neglected [28]. The new attack space of CPS varies due to the discrepancies in their structure, platform, and applications. Thus, many of the well-known methods are not appropriate for CPS, as they do not handle correlated features in sparse and high dimensional space, differentiate noise from the original behavior, and are not applicable to stream data [301].

Besides, anomalies can originate from various sources like cyber or physical attacks, errors in control or communication networks, and other malfunctions. Differentiating *anomalies of interest*, suspicious anomalous behavior indicating a potential security threat, from the set of those anomalies considered irrelevant to security, is often an even more intricate task than finding anomalies [260].

5.4 Proposed Method

AttackTracker is a scalable framework applicable to supervisory control systems. As Figure 5.1 illustrates, a hierarchy of *attack detectors* continuously monitor the operation of controllers at different levels of a control system. (l_1)-detectors monitor peripheral controllers such as PLC units at the local level. While, at higher levels, (l_i), $i = 2, 3, ...$, detectors monitor the output of multiple detectors at level (l_{i-1}). At the top level, a single detector determines the global operational status of the entire system and reports attacks in progress in any one of the subsystems. A local detector continuously analyzes the operation of a subsystem as reflected by the state of its sensors and actuators to spot deviating observations in the data stream that form a collective anomaly associated with an attack targeting this subsystem. Each local detector consists of a Behavior Predictor (Section 5.4.1) feeding the 'expected' next observation values into an *inference engine* (see Section 5.4.2). The inference engine processes and labels observations, assigns attack scores, and raises red flags based on the deviation of observed values from predicted ones relative to a dynamically adjusted threshold. For $i \geq 2$, (l_i)-detectors aggregate data and information received from their lower-level detectors to identify the attack scope in the underlying levels. This way, detectors operating at the higher levels are able to distinguish distributed threats in addition to centralized attacks.

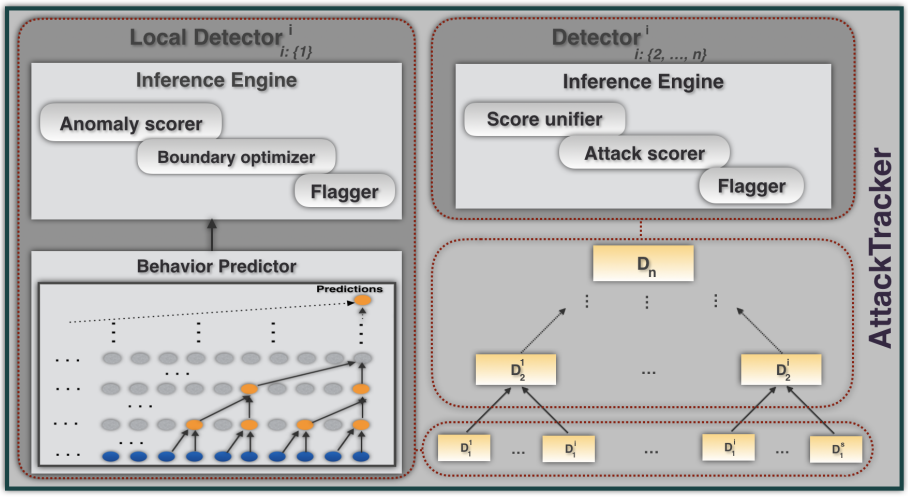


Figure 5.1: AttackTracker framework architecture

5.4.1 Behavior predictor

A behavior predictor learns hidden patterns from a history (X_T) of discrete observations in the form of a multivariate stochastic time series in order to predict the next observations X_{T^+}, where $T^+ = \{t+1, ..., t+l \mid l \geqslant 1\}$, with a reliable accuracy, meaning the error deviation is limited with a symmetric distribution. In the AttackTracker framework, each Behavior Predictor (BP^s) uses a customized mutivariate TCN (MTCN) model [21] to learn the normal behavior of a subsystem and forecast the next local feature values (x^s_{t+1}) based on the previous observations $\{t-w, ..., t\}$, where w defines the length of the lookback window.

$$BP^s(x^s_{t-w,...,t}) = x^s_{t+1}, s \in S \tag{5.2}$$

To be suitable for time series forecasting, TCN models apply a unique convolution, called *causal convolution*, where inputs can only be connected to future time-step outputs in a causal structure. Moreover, taking advantage of *dilated filters* and *stacking the convolutions* [21], empowers TCN models to handle the entangled properties with real-world CPS signals and find latent patterns:

- *Long-term events*. The scale and duration of latent patterns are highly different from one another, depending on seasonality, cycles and other independent stochastic factors. Therefore, a profiling method should be able to memorize observations and complex patterns occurring over a rather long historic window. There are two essential features which make TCN suitable to fulfill this requirement, while it converges faster with higher accuracy than other methods such as RNNs:

  - *Dilated filters* provide the possibility of covering a larger area of input data by skipping fixed steps between every two adjacent selected points for the next level. Therefore,

based on the convolution layer depth, the neuron's receptive field will exponentially increase.

  - *Stacking the dilated convolutions* increases the reception fields, while it preserves the input resolution and computational efficiency.

- *Noisy environment*. Signals may include unrelated noise because of sensor faults, noisy networks, or other external factors. Hence, the profiling method should be able to avoid being overfitted to noise, when learning the underlying hidden patterns. Because of the *dilated filters* property, a TCN is able to skip noise and memorize reoccurring patterns in different levels of granularity.

Besides increasing the scalability of the framework, having a hierarchy of local predictors helps to accurately learn the behavior of each individual subsystem.

5.4.2 Inference engine

Cyberattack detection is going beyond the profiling of normal behavior and labeling any violation from the expectation as an anomaly. The main aim of AD is finding the isolated points or events not been observed before, like real malicious events, or any kind of noise or benign events [260]. Therefore, AttackTracker should filter the unrelated anomalies to mitigate the false alarm rate, while keeping a very high accuracy level. Thus the *inference engine* is responsible for:

- ***Anomaly scoring***. Assigning proper scores that imply the likelihood of an event to be anomalous in comparison to the previous observations [91].

- ***Boundary approximation***. Finding a suitable *isolation boundary* to differentiate potential suspicious events from normal ones, based on available information and observed discrepancies in the train and validations sets.

- ***Flagging***. Analyzing the collected scores to filter out insignificant outliers and flag the collective events in case their behavior is suspicious.

Moreover, there are two major concerns that the *inference engine* is supposed to address: 1) Data-driven predictors may suffer from abrupt unsteady results potentially caused by temporal predictor faults, or outliers due to noise or data drift; 2) Each predictor is learning the subsystem behavior and is able to address potential feature correlations, but is oblivious to the subsystems handled by sibling detectors (other detectors at the same hierarchy level). Therefore, AttackTracker is distinguishing real threats by applying a strategic aggregation and prioritization of alarms obtained from the network of detectors, considering available contextual information and subsystem correlations.

Individual scoring

This phase will be performed in two steps, one offline and one online. The offline step prepares the decision-making logic of the inference engine, while the online step automatically detects the suspicious events.

Offline step. The raw error vector $\bar{E}_s$, with $\bar{E}_s : e_t^s = |y_t^s - y_t'^s|, s \in S$ for S local subsystems, is normally not a reliable source for anomaly detection due to several reasons such as possible deficiencies of the fitted model and the variance of its quality over the entire feature space. Moreover, the results of predictors (BP^s) across local detectors (in l_1) refer to unlike scales. Each *inference engine* applies the modified z-score (Z^s) [138] on the divergence error ($\bar{E}^s$) from the real-world observations to make them compatible for *anomaly scoring*. The modified z-score applies *median absolute deviation (MAD)* and *median*, which makes it more reliable in the presence of outliers compared to the standard z-score.

$$\bar{Z}^s : z_t^s = 0.6745 * (e_t^s - \text{Median}(train)^s)/\text{MAD}(train)^s, e_t^s \in \bar{E}^s \tag{5.3}$$

Also, this way of scaling is highly insightful and interpretable because of presenting the divergence degree of each single observation (x_i^s) based on the number of *MADs* that it is far from the expectation.

Then, to find the *isolation boundary* a log-based threshold (*Thresh*) of anomaly scores will be computed based on a normal reference set (*Ref*) which has not been observed by the Behavior Predictor. To loosen the threshold-based filtering, considering the potential trends in the near future, a confidence interval, called *slack*, is added to the threshold; the *slack* value is the maximum difference in z-scores of the reference set in comparison to the train and validation set.

$$Thresh^s = max(|MAD^s_{(Ref)}|) + slack^s \tag{5.4}$$

$$slack^s = max(|MAD^s_{(Z^s(Ref))} - MAD^s_{(Z^s(valid))}|, |MAD^s_{(Z^s(Ref))} - MAD^s_{(Z^s(train))}|) \tag{5.5}$$

Online step. Based on the knowledge extracted in the offline step, the online step is focused on *flagging* the suspicious events of local systems. Attacks aimed at severely disrupting or damaging the system, usually cause cascading effects which may impact the system behavior for a while, meaning, an important indicator for filtering out unrelated noises is the duration of anomalous events. Hence, a persistent anomalous interval is more suspicious of being an attack than a single strike caused by sensor noise or predictor faults, while the rate of divergence matters. Thus, AttackTracker considers a novel approach to perform the dynamic flagging based on the tradeoff between deviation and persistence. To this end, it computes the moving average ($\bar{M}^s$) of absolute z-score vectors ($\bar{Z}^s$) within the temporal window (w_p), selected as the average number of steps that an anomalous event is typically expected to last to be associated with an attack. This decision-making strategy helps AttackTracker to identify collective and correlated anomalies as one single attack.

$$m_t^s = m_{t-1}^s + (z_t^s - z_{t-w_p}^s)/w_p \tag{5.6}$$

Thus, the first observation (x_i^s) with an m_t^s value higher than the threshold will trigger the system to raise a red flag for a local attack, provided that at least 50% of the points are detected as anomalous in the considered w_p, which is the direct predecessors of x_i^s. The second condition is an offset to mitigate the influence of unrelated massive strikes. If the system raises an attack flag under this condition, it keeps the system in the same status as long as the listed conditions persist. In other words, the system status will go back to normal as soon as one of the two aforementioned conditions disappear.

The window length should be short enough to warn the reactive controllers as soon as possible so that immediate actions be taken to stop potential cascading effects of an attack on the other critical subsystems. For example, in the SWaT testbed as the dataset applied in this study, there is a sequential and a circular dependency through Stages 3 to 6. As Figure 5.6 illustrates $SSSP_8$ is an attack on DPIT of Stage 3, which spreads to Stages 4 and 5 after a considerable lag; so, the persistency should be customized based on the system properties and its cascading lags.

System-wide scoring

Since complex CPSs include interconnected subsystems, the higher level detectors ($(l_i), i = 2, 3, ...$) efficiently aggregate the results obtained from individual detectors to point out all the system-wide attacks. For example, there are situations where the whole network is under attack but the individual detectors are unable to detect this scenario because of losing their trace in the distributed nature of the network. Further, having simultaneous anomalies pointed out by more than one detector is another strong indicator of potential threats. For this reason, we need to calculate the global log-based anomaly scores. The z-score scaling does not guarantee that the local processes follow the same distribution; in fact, their range of thresholds and score variations may vary considerably. To illustrate this aspect consider Figure 5.2, which shows the result of the local detectors belonging to the 3rd and 4th stages in a short interval of test streams.

As shown, the scores assigned by local detectors to the same events are hugely disparate, because of different levels of sensitivity to the events. Moreover, some attacks might be missed by one or another due to not being aware of the global system view. To unify these values, each detector (l_i) computes the rate of anomalies ($\bar{R}^s$) in its underlying subsystems ($l_i{}^s$) as the scaled value of anomaly scores by their own threshold ($r_t^s = z_t^s / Thresh^s - 1$), so that the scaled scores higher than zero represent the system-wide anomaly scores $\bar{A}$:

$$\bar{A} : a_t = \begin{cases} max(r_t^{1..s}) & \text{No anomaly subsystems} \\ \sum_j (r_t^j), r_t^j > 0 & J \text{ out of } S \text{ are anomalies} \end{cases} \tag{5.7}$$

Finally, the inference engine at the global AD level uses a strategy similar to the one used by local inference engines. A red flag for a system-wide attack will be raised for observations with a moving average, computed based on a system-wide score, higher than zero ($m_{t,w_p} > 0$).

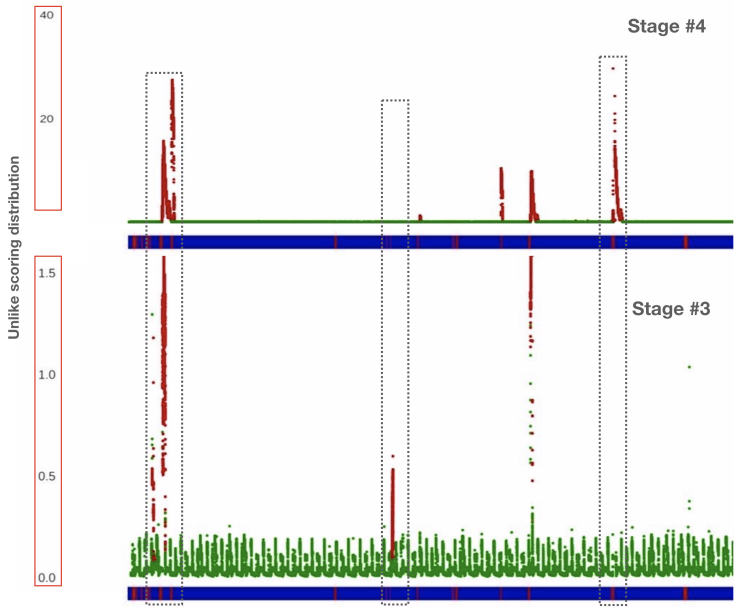


Figure 5.2: The difference of scoring in the 3 and 4 local detectors

5.5 Data Characteristics

In this study, we target the fully operational secure water treatment (SWaT) testbed [109] provided by Singapore University of Technology. The SWaT data has been collected from a complex real-world facility targeted by a variety of realistic attacks on different parts of the system[1]. We briefly describe here the testbed's general properties and refer the reader to [109] for further details

The main goal of this plant is to supply drinking water purified through a six-stage process, comprising control and physical components. Each stage includes 1) sensors monitoring the water level, flow meters, conductivity analyzers, PH analyzers, pressure meters and more; and 2) actuators for operating valves to control the inflow and different types of pumps. The treatment process starts in P1 by taking in raw water and storing it in a tank. Then the water quality assessment and chemical dosing are done in P2. Subsequently, P3 removes undesirable materials and any remaining chlorine is destroyed in P4. Next, the water from P4 is pumped into P5 to reduce inorganic impurities. Eventually, P6 stores the treated water for distribution in a water distribution system.

[1]There is limited availability of operational datasets in the field of securing CPSs. Even though there are a few datasets, such as DARPA Intrusion Detection Evaluation Dataset [182] and the NSL-KDD99 [23] for IDSs, these datasets are: 1) Primarily focused on network traffic and not suitable for CPS IDS. 2) Most of them are easy datasets and do not face any challenges (even redundant records). 3) Many of them are synthetic datasets and cannot reflect real-world behavior. Also, there is another set of publicly available datasets for CPSs provided by the Critical Infrastructure Protector Center at Mississippi State University. However, these datasets are found to contain some unintended patterns that can be used to easily identify attacks versus non-attacks using machine learning algorithms [109]. Hence, to the best of our knowledge, there is no other publicly available realistic dataset of sufficient complexity from a modern CPS that contains both network traffic data and physical properties of the CPS.

Control units interoperate with the system through a layered communication network. Programmable logic controllers (PLC) read sensors and trigger actuators at the local layer in each of the six stages; while the global layer combines the local views using a supervisory control and data acquisition (SCADA) system linked to each of the locally operating PLC units. In the data collection process, only network data through wired communications is collected.

The data was collected over 11 consecutive days of continuous operation, one data point per second. The recorded logs represent normal operation over the first seven days. During the following four days, the system was targeted by 36 different attacks. Table 5.1 lists four different types of attack scenarios, each lasts from a few minutes to an hour, targeting one or several points in a single or multiple stages. The scenarios are referred to as *single point single stage* (SPSS), *multi point single stage* (MPSS), *single point multi stage* (SPMS), and *multi point multi stage* (MPMS), respectively.

Att. cat.	**freq**	**Targets**
SSSP	26	MV_{101}, MV_{504}, P_{102}, P_{102}, LIT_{101}, AIT_{202}, LIT_{301}, $DPIT_{301}$, FIT_{401}, MV_{303}, MV_{304}, AIT_{504}, P_{302}, LIT_{401}
SSMP	4	(P_{203}, P_{205}), $(P_{201}, P_{203}, P_{205})$, (LIT_{401}, P_{401}), (MV_{101}, LIT_{101}), (P_{101}, P_{102}), (P_{501}, FIT_{502})
MSSP	2	(P_{101}, LIT_{301}), (P_{302}, LIT_{401}), (AIT_{402}, AIT_{502}), (FIT_{401}, AIT_{502})
MSMP	4	$(P_{602}, DIT_{301}, MV_{302})$, $(UV_{401}, AIT_{502}, P_{501})$, $(LIT_{101}, P_{101}, MV_{201})$

Table 5.1: Attack types and their target points

5.5.1 Dataset Exploration

We performed a comprehensive data exploration and analysis on the SWaT dataset to learn about its statistical characteristics, feature distributions, and their stability. In this section, we examined some aspects of the available test set; however, to adhere to a semi-supervised approach, we did not use the knowledge obtained from the test set in the method section. Also, since the data collection has been started from an empty state and the system shows remarkable instability in the first few hours, we exclude this part of the training set. Further, we apply two basic outlier filtering techniques of fast Fourier transformation (FFT) and rolling mean on the random features to remove short-term noises.

Invariant variables. The training set includes ten actuators, mainly backup feeding and dosing pumps, the status of which is steady; we exclude, these zero standard deviation features from our predictive model because the behavior predictor will predict the same constant values for them. However, our system will continuously monitor this set of features and logs any anomalous event that triggers a change in their constant status. For example, a few attacks in the SWaT dataset affect the NaCl and HCl levels, which in turn affect the settings of the corresponding dose pumps (P201, P204).

Feature distribution. Assuming that unseen data originate from the same distribution as the training data is a typical pre-requisite for most predictive machine learning models, since unfore-

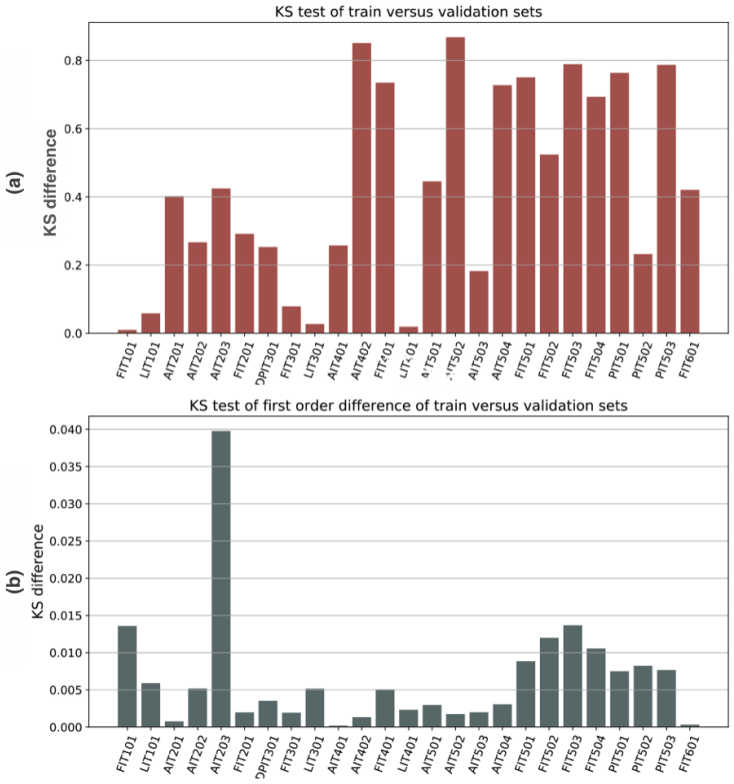


Figure 5.3: (a) KS test result on scaled feature values of train vs. validation sets, (b) KS test result on scaled first order difference of train and validation sets

seeable statistical property changes may hugely influence the final results. A common method to evaluate the similarity of the training and test distribution is KS test[2] [153, 244], however, since this method needs to observe the whole test set beforehand to prune the drifted feature sets, it is not appropriate in an online setting stream analysis context. To make sure that the system behavior is predictable over the course of time, we have applied the KS test on the training and the validation set. As Figure 6.1 (a) illustrates, the KS difference of many of the features is notable with significant p-value. But, statistically speaking, this test is sensitive to differences in both location and shape of the cumulative distribution, so part of this difference is caused by incomplete temporal dependencies. Thus, we applied the Augmented Dickey-Fuller test[3], which confirms that most of the features are non-stationary. In the next step, Ljung-Box test[4] revealed that they are highly auto-correlated. Hence, we applied the KS test on the scaled first order difference of the original features, which removed

[2]The Kolmogorov-Smirnov (KS) test is a nonparametric test of the equality of one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution, or to compare two samples.

[3]Augmented Dickey-Fuller is a statistical test to verify the presence of a unit root in analyzing how strongly a time series is distorted by an arbitrary trend.

[4]LjungBox is a statistical test which evaluates if any group of auto-correlations of a time series is different from zero.

part of the auto-correlation. As Figure 6.1 (b) shows, the distribution of the transformed version of the train and test set are quite similar, with insignificant p-value. Thereby, the soundness of the train set features is confirmed and there is no significant drift in the predictors to be filtered out. Even when applying a principal component analysis (PCA) clustering on the dataset the result confirms that the validation set follows the same distribution as the train set. PCA is not able to partition these two sets from each other. While, as part (b) shows, the normal cases in the test set are simply separable from the train set observations.

Figure 5.4: (a) PCA clustering on scaled feature values of train vs validation, (b) PCA clustering on train vs normal cases in test set

Therefore, some of the predictors in the test set should be fairly different from the given train set. By way of illustration, consider the feature AIT201, a conductivity analyzer which measures the NaCl level. As Figure 5.5 shows it has a drastic dropping trend, which causes an entire dissimilar test set distribution. If we scrutinize the sequence of this predictor, this divergence starts right after attacks 19 and 20, which are targeting AIT504, hugely affecting the NaCl level. Such significant drifts of data may influence the result and generate too many false alarms. But, since the test set is not available in stream analysis, all the features have been included in this study and the applied method decides based on the overall picture of subsystems, not one specific predictor.

Correlated features with delayed cascading influence. SWaT has highly correlated features at the local and system level, some of them are based on causal relationships. For example, some of the features (sensors) from Stages 5 are highly correlated. To avoid multicollinearity, excluding the highly correlated features is a general practice in statistical models, but there are two major drawbacks that make this strategy undesirable in our context. First, if a filtered feature is under ongoing attack, the model might be able to detect the threat based on the other correlated features, but would not be able to point at the targeted features; this problem is more severe in high-dimensional datasets.

Second, since we are presuming that the majority of the available data in the training phase is normal, we are not sure how the attack may impact the system behavior. In other words, there is a slim chance that an attacker would be able to correctly detect all the other correlated features and force them to stay within the normal range. Hence, to capture the potential attacks in the fastest time

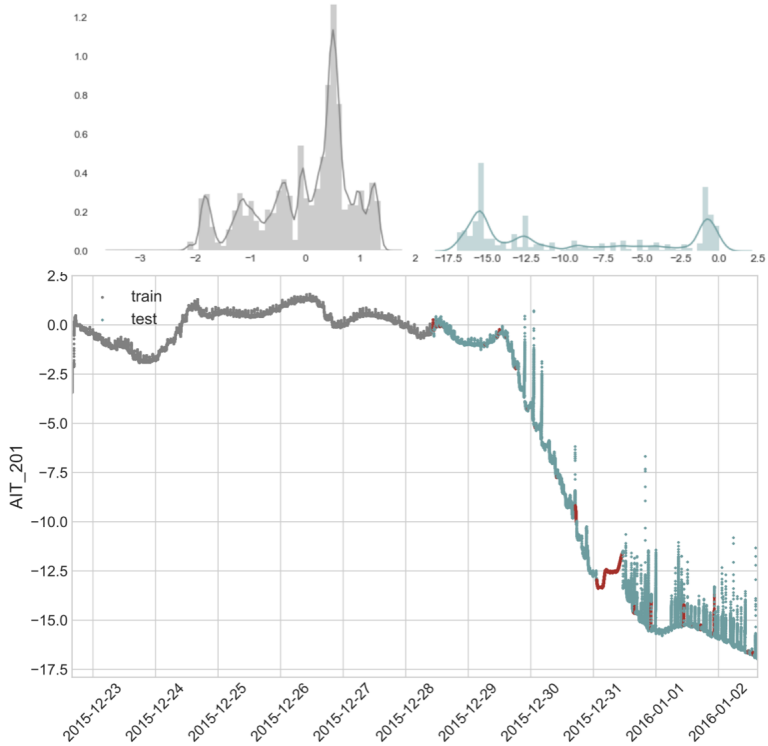


Figure 5.5: Feature AIT 201 in train and test set with red subsections as attack

possible, simultaneous monitoring of all the signals is required. However, to mitigate the false alarm rate, the detector model should be advanced enough to handle the latent correlations.

As an example, attack #28, presented in Figure 5.6, shows that a few seconds after P302 is closed by attackers, the DPIT301 status changes, but its influence propagates to the subsystems 4 and 5 with a delay of about 35 minutes. While, when considering the same targeted stage in attack #23, the DPIT301 value is affected, but the problem does not cascade through the system because the duration of the threat is shorter than the cascading delay interval.

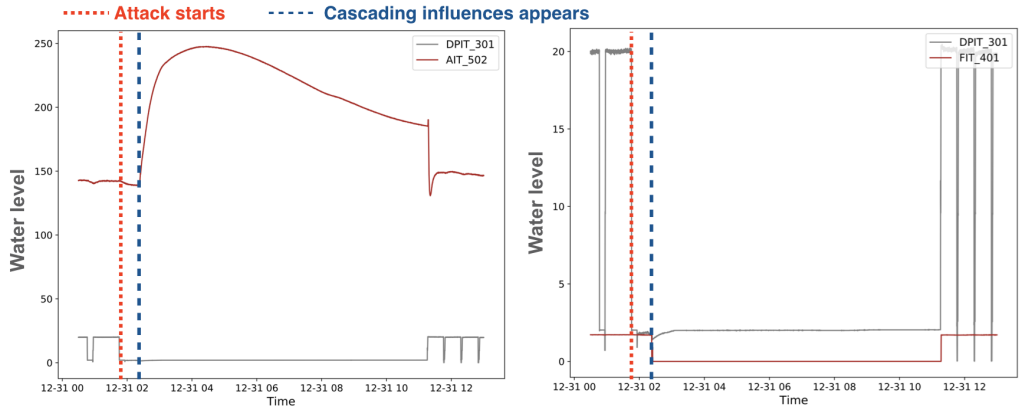


Figure 5.6: Delayed cascaded attack impact in the subsequent stages

Long-term Recovery Phases. Due to inter-system dependencies and the impact of events on the system, after each attack, it takes a while for the system to recover and return to its normal operation. For example, as Figure 5.7 illustrates, during the recovery period for two different attacks, $MSMP_{30}$ and $SSSP_8$, with $\{LIT_{101}, P_{101}, MV_{201}\}$ and $\{DPIT_{301}\}$ respectively as attack targets, the system is not behaving normally and AttackTracker may keep the red flag on. While this is correct, it also causes false positives (FP) in the log-based analysis. Not raising a red flag in the recovery phase can be very challenging. The fact that some of the attacks have been launched when the system was unstable and recovering from a previous attack potentially increases the complexity of our problem.

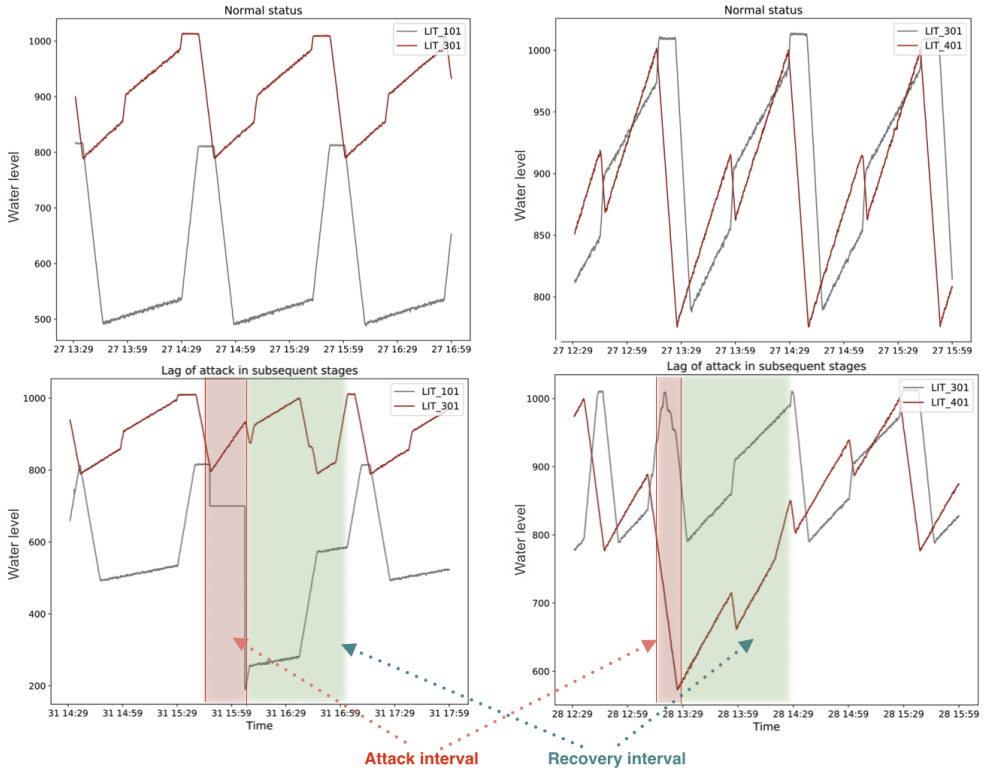


Figure 5.7: Real examples of recovery interval

Unsupervised clustering. To determine the separability of under-attack intervals from the normal operation, we applied t-SNE on the raw and window-based sequences of this dataset. Figure 5.8 represents two principal t-SNE dimensions[5] of attacks versus normal data points. Some of the attacks, depending on their feature values and their correlation, are easily separable from the normal cases, while most of the tricky cases are tightly interwoven and obscured in the normal observations. Being highly dependent on various external factors, the observed system behavior is stochastic, which makes finding potential attacks a tricky and challenging task requiring a customized analysis method.

[5]t-distributed stochastic neighbor embedding (t-SNE) is a non-linear technique for dimensionality reduction that is for the visualization of high-dimensional datasets.

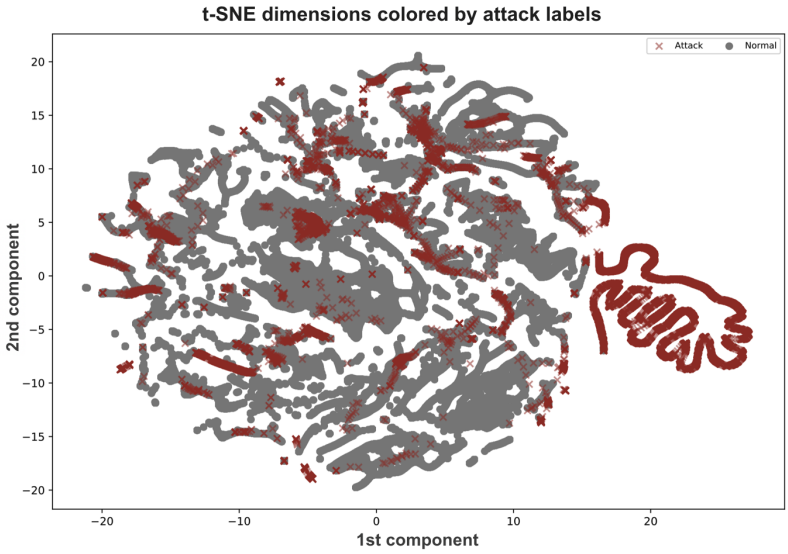


Figure 5.8: A t-SNE of normal and attack observations with attacks highlighted in red

5.6 Experiments

In our experimental evaluation, we compare AttackTracker to the other research performed on the SWaT testbed. Since none of the techniques were fully reproducible or available to apply, the results reported in the corresponding literature are considered as peer method's performance. And, as the other studies, we have ignored the attacks which have no impact on the physical system behavior.

Experimental settings. In our experiments, each of the six local SWaT stages is considered a separate subsystem. We follow a 2-level hierarchy and train separate MTCN models on each local stage to predict the upcoming values of sensors and aggregate the results in the system level, called AttackTracker_2. However, to find the best level of granularity for the whole system, two other model configurations are also analyzed: 1. AttackTracker_1: the whole system is considered as one subsystem and one Behavior Predictor is trained; 2. AttackTracker_3: Each feature is considered as a subsystem and the model is trained at three hierarchy levels. Further, to verify the influence of our proposed scoring and labeling method, the attack detection experiments use a modified version of the three aforementioned settings, called MTCN_1, MTCN_2, and MTCN_3, which apply a basic scaled prediction error and mean-based aggregation to detect attacks. As aforementioned, the SWaT signal set has been used in several cyberattack detection studies, the reported performance of which we compare to AttackTracker. Further, the Behavior Predictor hyper-parameters are optimized based on the loss value of the first validation set. The optimization process is based on a grid search, so potentially there is room for further improving the behavior prediction results. The lookback window size (w) is 100 and the w_p is set to 40. Some of the fixed hyper-parameter implementations across all of the MTCNs include: kernel sizes are set to the constant value of 8 with a filter size of 24 to avoid being very local; dilation factors are d = 1, 2, 4, 8, 16. Model parameters are optimized using the RMSE loss value through Stochastic Gradient Descent and ADAM step updates and the activation

functions are normalized ReLU. Last but not least, two transformations are applied to the input data. The first-order-difference is computed to remove the trends and varying means. Then, these values are normalized (using a standard scale) to stabilize the variance.

5.6.1 Behavior predictor evaluation

First and foremost, we evaluate the performance of Behavior Predictor to show the reason behind applying MTCN. To this end, we utilize the two quadratic error measures of root mean square error (RMSE) and mean absolute error (MAE) as the success criteria for the comparison, which use the full-spectrum of the outcomes. As Table 5.2 shows, MTCN outshines multivariate LSTM, support vector regressor (SVR), and random forest regressor (RF) by achieving 0.006 and 0.004 as RMSE and MAE, respectively.

Table 5.2: Behavior Predictor performance for standard scaled values

Mode	Train		Validation	
	MAE	**RMSE**	**MAE**	**RMSE**
RF	0.012	0.016	0.013	0.016
SVR	0.007	0.008	0.009	0.013
LSTM	0.003	0.004	0.006	0.009
MTCN	**0.002**	**0.002**	**0.004**	**0.006**

5.6.2 Log-based evaluation

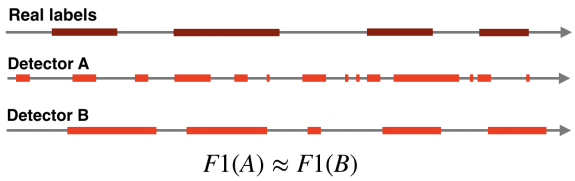


Figure 5.9: An example of a logical detector vs. a fluky detector

This subsection presents the log-based evaluation of AttackTracker's performance in comparison with the related studies and peer methods. The evaluations are performed based on binary labels because the unified anomaly score of the ground truth is not available.

As Table 5.3 shows, AttackTracker outperforms the other state of the art techniques by achieving an outstanding F1-score in the presence of a very acceptable recall and precision tradeoff. Another key point to note is the difference in performance in the series of MTCN_i detectors compared to their respective AttackTracker_i, which highlights the influence level of the *Inference engine* algorithm. This improvement is mainly due to the persistence in the scoring of computations, which filters out the potential short-term spikes caused by noise or model errors. It is worth noting that the authors

of [161] report a very high F1-score based on 1D-CNN. However, the reported scores in their study are not log-based and are computed based on the attack entities, similar to the evaluation approach presented in the Section 5.6.3, but on the basis of false positive (FP) and true positive (TP) metrics. They also ignored the recovery windows in the recorded result, which can hugely influence the reported scores.

Table 5.3: The log-based performance of AnomalyTracker

Model	**Precision**	**Recall**	**F-measure**
TABOR [181]	0.862	0.788	0.823
DNN [139]	0.983	0.678	0.803
one-class SVM [139]	0.92	0.699	0.796
$MTCN_1$	0.709	0.917	0.801
$MTCN_2$	0.64	**0.986**	0.789
$MTCN_3$	0.691	0.982	0.811
$AttackTracker_1$	0.91	0.813	0.859
$AttackTracker_2$	**0.947**	0.855	**0.898**
$AttackTracker_3$	0.869	0.843	0.856

5.6.3 Event-based evaluation

AD-based cyberattack detection is beyond anomaly detection and there is no value in achieving a very high precision and recall just by chance and dispersed labeling of observations because security threats are mostly related to collective and correlated anomalies. As Figure 5.9 illustrates, a hypothetical detector, called *A*, could attain almost the same F1-score as *B* by coincidence. Obviously, the fluky detector *A* generates many scattered false alarms over the time span, which are not desired; while detector *B* systematically chases the event-based correlated and collective anomalies in order to raise significant and intuitive alarms. Besides missing the collective aspects of attacks in analyzing the labels assigned by AD methods, log-based evaluation is not appropriate in an unbalanced context, which is the basic property of attack-based labeled datasets like SWaT. Moreover, when small randomness in results is unavoidable or close-calls are possible, which is highly probable in AD contexts, distinct labeling may not be a reliable source to measure the quality of one method versus another. As aforementioned, the recovery of subsystems and features under attack may take some time. If the attack detection system properly keeps the red flag on during this period, that may result in a high volume of FP in the log-based analysis.

Hence, in [181], event-based FP is defined as a detected subsequence without overlapping with any attack scenarios; they analyze the TPs based on two detailed metrics, *coverage proportion (CP)* and *penalty score (PS)*. CP evaluates the quality of detection coverage based on the fraction of overlap and the total length of ground-truth scenarios (higher is better), while PS indicates the length of detection outside the overlapped interval (lower is better).

We evaluate AttackTracker's delay in detecting attacks according to another metric, referred to as *average of delay ($\bar{D}$)*, which is a highly crucial factor in real-time AD but is missing in any of the other measures. As the obtained results demonstrate, our method outshines TABOR in CP and PS, while raising only a single false red flag. AttackTracker's delay, $\bar{D}$, is 18 seconds; TABOR's delay is not mentioned in [181].

Table 5.4: Event-based performance of AnomalyTracker

Model	**FP**	**CP(%)**	**PS(s)**	$\bar{D}$**(s)**
TABOR [181]	**0**	19.5	831	NA
$MTCN_2$	7	**61**	1543	**4**
AttackTracker_2	1	37	**342**	18

5.6.4 Discussion

The high quality of AttackTracker has several principal reasons: 1) It utilizes a vastly accurate Behavior Predictor, which is able to precisely forecast subsequent observations in multivariate settings. To the best of our knowledge, our experiments, for the first time, demonstrate the forecasting capabilities of multivariate TCN, in the presence of highly correlated features. 2) It utilizes a novel inference engine for dynamic attack scoring. The anomaly scoring and labeling process has been overlooked in most profiling-based AD studies, however the benefits of systematic anomaly scoring for improved accuracy is exemplified by time-wise anomaly scoring of data points used by AttackTracker. 3) By utilizing a hierarchical structure, it can trace the attacks at different levels of granularity. While each local detector learns the detailed behavior of the subsystems, aggregation of their status provides a bird's eye view of the system situation. Another applied contribution of our studies is improving the accuracy level of cyberattack detection on the challenging SWat testbed.

We are aware of the fact that the results of our study may have certain limitations as follows. First, as for generalizability, one may argue that the experimental results obtained on SWaT may not carry over directly to other types of critical infrastructure and CPS because most of the anomalies in SWaT are arising from injected attacks and the AttackTracker's performance has not been verified for other types of attacks. We believe, not making any specific presumption about the underlying dataset or the type of attacks dissociates AttackTracker from specific system characteristics. Still, we realize further experiments are needed to verify the performance of AnomalyTracker on a wider range of real-world CPS and IoT datasets to cover additional types of attacks and anomalies.

Second, the current average attack delay is 18 steps, which might not be practical in some highly critical application domains. As we argued, in the context of the SWaT testbed and probably many similar CPSs, AttackTracker can be considered to operate in near real-time because of its cascading delay tolerance. Although the delay can be decreased by tuning the model, we aim to improve the scoring technique toward decreasing the delay in attack detection without losing the precision level.

Also, we are planning to enhance AttackTracker's performance in three other ways: 1) Improving the interpretability level of the results by highlighting the attack target and its potential cascading influences. This insightful information helps the end-user to choose the best mitigative action. 2) Boosting the Behavior Predictor component to find potential drifts in the stream data and adapt itself so as to not being fooled by attacks. 3) Enhancing higher level detectors by utilizing Behavior Predictors to trace the collective behavior of their underlying subsystems. Accordingly, an inference engine should decide based on a multi-modal view provided by its associated Behavior Predictor and the underlying detectors.

5.7 Conclusions

Behavior-based intrusion detection complements signature-based detection in dealing with increasingly common zero-day exploits. AttackTracker presents a novel, distributed online attack detection framework for processing supervisory control signals from the continuous operation of cyber-physical systems. Dynamic attack scoring boosts the analytic performance to efficiently detect collective attacks by orchestrating a network of detectors and reduces the false alarm rate by ignoring potential contextual noise and errors in the Behavior Predictors. Being able to handle regular spikes of observed 'anomalies' in cases where Behavior Predictors have not captured all the patterns of normal data is another advantage of our framework.

AttackTracker significantly outperforms the other state-of-the-art methods and studies applied to the SWaT testbed by sustaining a high rate of precision and recall, while achieving a 7% log-based F1-score improvement. Further, it raises only one false alarm event, while it is able to find all of the attacks that physically impact the system. Cybersecurity research is a principal area of innovation to protect our economy and enhance the security of societies worldwide. Building a cyber situational awareness platform that addresses vital elements of physical and cyber threats to critical infrastructure is a crucial problem that cannot be overlooked. We believe the proposed intrusion detection framework opens new perspectives for advanced cyber-physical security, not only in the operation of common and widely deployed critical infrastructure but also in other applications of IoT systems.

Chapter 6

Anomaly Detection Explanation

High-performing, complex, black-box models are prevalent and inevitable. Despite the outstanding successes of advanced methods in many areas, there are some critical doubts about their precision when they are exposed to adversarial attacks. In particular, assigning nonzero weights to useless features or out-of-range values, deep learning models are known targets of attackers. Besides, achieving better numerical results without presenting the required intuition behind the applied decisions is not a sufficient and worthy contribution. Acknowledging that such techniques are not transparent and fully mature, and their transition might be timely, developing explainable algorithms to clarify the decision-making logic of such complex models is warranted, especially in large-scale and/or critical deployments.

However, it is highly challenging to explain temporal anomaly detection techniques. On the one hand, it faces the difficulties of anomaly detection, such as ambiguity and unclear definitions [299]. On the other hand, extracting patterns from highly auto-correlated and stochastic temporal sequences adds to the complexity of the problem [56, 298]. Furthermore, if the understanding stemming from a given explanation is not in any sense a true representative of existing affairs, then the explanation is misguiding and is not genuine. Therefore, this section describes a set of our researches that target the problem of temporal anomaly detection explainability from different perspectives. In Section 6.3, I propose a fully model-agnostic framework to explain any arbitrary model. In the next step, Section 6.4 is focused on explaining the autoencoder's results. Finally, in Section 6.5, I briefly explore the idea of a causal explainer on the basis of a causal profile discovery framework.

6.1 Introduction

Consider a multidimensional dataset of tabular features, such as signals continuously recorded by sensors in a highly interconnected system, which contains data points labeled as anomalous by an arbitrary black-box detector: (1) what is the best way to describe these anomalies and their differentiating factors to a human, and (2) how can the results be simultaneously succinct and effectively interpretable?

Anomaly detection (AD) has a rich history within the data mining and statistical machine learning communities [35, 205]. Various advanced standalone and ensemble time-aware, AI-based AD methods that learn data patterns and their evolution over time to precisely locate anomalous events in their temporal context with the least possible rate of false positives have recently been developed [50, 298]. Such key methods are urgently required for various event-sensitive scenarios such as causal disease discovery, robotic system monitoring, heterogeneous networks, and data center security [47, 152, 300]. However, the increasing diversity of data sources and associated stochastic processes are making this pivotal topic in data analysis more challenging than ever.

Considering existing challenges and vulnerability to adversarial attacks, AD should be transparent and verifiable to justify trusting the outcome. Thus, a detector is expected to build trust by involving the human analyst in the predictive process. To this end, a model should provide meaningful insight into its decision-making process and reasoning based on its input features. Meanwhile, not clarifying the internal behavior of complex black-box models makes them appear mysterious to the user; such models are doomed to be abandoned even if only a few unpredictable and inexplicable mistakes occur. For example, it is not easy to interrupt or shut down a critical control system just because a normally accurate AI model predicts some unknown, enigmatic risk or threat. This challenge could be even more difficult if the user needs to take action but is unsure of the risk factor. This is why most advanced AI methods, such as deep neural networks that learn from knowledge and extract complex patterns without any explicit programming, are intractable and uninterpretable black boxes [78]. In other words, they imprison knowledge and intuition and only offer the final results.

In the last few years, the topic of explainable artificial intelligence (XAI) has been highly appreciated in AI communities and various strategies, frameworks, and measures have been proposed to explain or assess the interpretability of black-box models [78]. Given that the black-box nature of such complex AI-based techniques is far from being interpretable [16], wrapper explainers, also called post-hoc explainers, are considered to bridge black-box machines and humans. This matter has been performed through rule-based, mathematical formalizations, feature attribution, visual explanations, etc. [16, 78]. Such explanations can be local or global [78], dependent on a specific model [257], or fully independent of the black-box [235].

While there is considerable prior research on both XAI [16, 120] and AD [50, 298], the literature on AD interpretation is comparably sparse. An XAI model for AD should be able to handle complex challenges of the open-ended definition of anomalies [128] and the fundamental ambiguity about the actual nature of an anomaly and its potential boundaries with the normal events. Anomalous events might originate from heterogeneous statistical or structural sources, result from various causes, and occur in very different and even difficult-to-predict ways [176]. Moreover, such contexts require an unsupervised or semi-supervised technique, which might be extremely challenging or impossible for the existing XAI techniques as a labeled training set is not available for their explanation process.

Given that many advanced AD models utilize sequential or temporal aspects [298], their feature space is highly autocorrelated and too complex to be explained by generic and model-agnostic XAI

methods [121, 235, 249]. In addition, increasing the size of the temporal window provides the AI model with extra knowledge of data, but might cause a loss of information about the exact location of the occurred attack and decrease the model's interpretability level. This is because the alarm is not necessarily raised at the exact time of which anomalous event is observed. Also, if a model misses a few of the actual anomalous points due to inconsistency, it might generate false alarms for subsequent normal events.

Motivation. To the best of our knowledge, within the literature on anomaly explanation or description, there are no reports of model-agnostic XAI for temporal AD, let alone for explaining anomalies in multi-dimensional stream data. Most of the generic explainability frameworks are based on simplistic presumptions such as independent and identically distributed random (i.i.d.) variables, balanced datasets, linearity, and supervised settings, which contradict the challenging nature of anomaly detection in stream data due to various factors such as non-linear and non-stationary processes, latent confounding causes and potential noises.

6.2 Background

If an event with very rare properties is observed, an anomaly detection algorithm raises a red flag. The rarity of an event may be latent with respect to various properties or patterns associated with the feature values or their context. Nowadays, advanced methods such as complex deep learning models are powerful enough to discover such differences but are not crystal clear. Anomaly detection concepts face fundamental challenges such as complex latent patterns, concept drift, and overfitting that can mislead the model and cause a high false alarm rate [260]. Furthermore, the restrictive closed-world assumption [284], which only specifies the normal profile and considers anything else a negative case, can increase the chance of false alarms. Therefore, not knowing a model's internal process deprives the end-user of the ability to be familiar with the model's logic and trust it. Moreover, such obscurities restrict developers from addressing the model's potential vulnerabilities.

Another major barrier to applying advanced anomaly detectors in critical systems is their vulnerability to adversarial scenarios. Well-performing complex models might be easily fooled by very slight perturbations that are inappreciable to humans [212, 291]. Thus, such approaches may not be effective or reliable against various types of attacks.

General Background. Deciphering the black-box of machine learning models is one of the most challenging, top-priority, and critical problems. This is not a new problem; one of the earliest black-box puzzles goes back to Dean Pomerleau's 1991 tussle when teaching a computer how to drive. Complexity-wise, that problem was childish compared to the technique of deep learning in the big data era, which had broad applications in different contexts such as security systems, self-driving cars, and creative systems [46]. The purpose of explaining a system or model can vary depending on the type of the users addressed by the explainer. For instance, the work in [261] categorizes such purposes into five common types: transparency, learning, conceptualization, justification, and

relevance. This study focuses on the combination of *transparency* and *relevance*, but the results provided can also implicitly be applied toward justification of the system.

A few promising approaches have been proposed to improve the explainability of AI models. In order to deeply understand the key criteria and challenges for developing such an approach, we direct the reader to previous comprehensive studies [119, 120, 229] that have scrutinized the explainability context; herein, we review the major concepts and highly relevant techniques. The existing techniques can be classified according to several aspects. Concerning the coverage span, the explainers may be global or local. A *global explainer* provides a generic explanation using interpretable models [61] or rulesets [165]. Meanwhile, a *local explainer* interprets a specific decision and is faithful to the black-box model, although there is no guarantee that the same explanation holds true throughout the whole domain space or even the proximate neighborhood [272].

On the other hand, these techniques can be categorized as either model-dependent or model-agnostic techniques. The former is customized for a specific method—for example, attention models [20] or some other previously reported techniques [103, 194]. Being aware of the internal process helps to contribute to highly accurate and advanced explanations, although such techniques are inapplicable to any arbitrary black-box. Meanwhile, model-agnostic techniques can interpret an arbitrary AI model as a black-box and are transferable. Examples of such techniques are LIME [234] and LEMNA [121], which provide the set of essential features that influence the black-box models decision in the local area.

Early XAI techniques focused on defining the black-box models on the basis of common rules [36, 199, 273] or visualizing the most important features [206], such as visualizing Bayesian models using nomograms. Later on, modern techniques such as LIME [234] and SHAP [189] aimed to find the most important features in the neighborhood of a specific data point by applying iterative perturbations. Therefore, these techniques may suffer from being too sensitive to non-influential features, which can consequently lead to unreliable explanations [279] and fail to provide faithful and local interpretations.

Also, given its focus on the authentication problems, LIME is customized to handle binary decisions such as image processing; therefore, it considers an independent and identically distributed (i.i.d) assumption, which is not the case in most real-world scenarios. Moreover, LIME is unable to handle the "curse of dimensionality", and the surrogate model distributes the feature importance among all the features. On the other hand, SHAP [189] introduces Shapley values [189] from coalitional game theory, as the unique solution in the class of additive feature explanation methods. However, SHAP has some critical drawbacks: it is undesirably exponential and substituting the feature values with all previously observed values can cause a loss of the locality assumption.

Nevertheless, most of the aforementioned model-agnostic explainers are unable to handle anomaly detection in the stream analysis context for three reasons. First, unsupervised techniques are mainly focused on generative features instead of creating boundaries, which are used by the existing classification-based explainers (e.g., perturbation techniques). Second, they cannot handle

the unbalanced distribution of anomalies. Third, they ignore the impact of contrastive contexts on or potential correlations with the isolating factors in the provided interpretation.

Anomaly detection explanation. A great number of studies have aimed to improve the efficiency of anomaly detection algorithms, but explanations of the reasons behind the anomalousness of the detected instances and their distinguishing features have not received enough attention. To date, a few studies have investigated the explainability of the identified anomalies, especially in numerical applications, but they are not strong enough to explain an arbitrary semi-supervised AD model. One of the earliest studies in this context [158] investigated the type of knowledge required by the end-user to understand anomalies and their differences. Moreover, that study categorized anomalies as *strong* or *weak* according to distance-based algorithms.

A Fisher linear discriminant classifier is applied by [64] to fully separate an anomaly from its neighboring samples determined using quadratic entropy. Another study applied classification accuracy to interpret the result obtained from an anomaly detection model [186]. However, balanced training sets created by generating artificial samples around the anomalous point were required. Meanwhile, an exceptionality score [13] is defined as a measure that highlights the extent of the difference in the feature distributions of outliers compared to those of inliers. Thus, this attribution can characterize the abnormality level of the given anomalous point or group. Based on the infrequency property of feature combinations in the outlier events, two main steps are taken to highlight outliers [13]: (1) generating a base condition, which includes conditions associated with each feature directing toward an anomaly, and (2) combining base conditions, which combines the conditions to form relevant explanations to represent a property of an individual. In other words, an infrequency property should be able to highlight the abnormality of an object if its combination of attributes would be far enough from the normal distribution.

COIN has been proposed as a detector-agnostic explainer that interprets an outlier on the basis of three aspects: its outlier score, the attributes contributing to its abnormality, and its contextual neighborhood [186]. This algorithm clusters the normal points in the anomaly's neighborhood and achieves interpretation by solving a series of classification tasks like LIME. The major drawback of COIN is that it performs synthetic upsampling to apply the classifiers; this may result in a loss of the correct isolating features. Sapling random forest is an ensemble of specifically trained binary classification trees that explains an anomaly on the basis of a subset of features in which the sample is the case that deviates most from the normal cases [159]. Sapling random forest recursively applies iForest and subsequently explains a specific data point by a conjunction of the atomic condition defined by the extracted features. LOOKOUT is another method that aims to present interpretable pictorial explanations of outlying behavior to the limited attention of human analysts [124]. This model outputs a few pictures from purposefully chosen feature subspaces on the basis of a plot-selection objective.

Meanwhile, model-specific attempts to explain differential temporal models, like attention techniques [134], estimating gradients of stochastic computations [87, 250], or reconstruction based methods [265] are another line of XAI research, which are only focused on deep learning models.

Despite the considerable performance of some of these techniques, their application is limited and cannot explain any arbitrary temporal AD model. In summary, most of the existing anomaly detection explainers configured for supervised settings, focus on explaining the anomalous observations distinguished by a classifier. However, this is not the case in the context of stream analysis using a semi-supervised model.

6.3 Dynamic Temporal Anomaly Detection Explainer

In light of the points developed in the previous sections, in this study, we formulate the explainability aspect of temporal anomaly detection in order to address the abovementioned challenges. In addition, we propose a framework for providing precise and simple explanations of anomalous behavior in multidimensional, real-world datasets concerning the sequential relations between stream data. The major contributions of the present study are as follows:

- The explainability problem of temporal collective anomaly detection, which is in demand in stream data analysis, is realized and formulated.
- The explanation algorithm reported herein is both domain- and detector-agnostic; furthermore, to our knowledge, this is the first model-agnostic study to explain anomaly scoring in temporal data.
- Case base and counterfactual reasoning were combined to highlight the isolation level of the input.
- The proposed solution can be internally applied to score anomalies; in this condition, it can be considered as an in-place explanation (i.e., an interpretable technique).

6.3.1 Problem definition

In this section, we propose a model-agnostic explainer that dynamically generates interpretable explanations for unsupervised temporal anomaly detection problems. In general, we assume a multivariate data stream $X = \langle x_t : t \in T \rangle$ interpreted as consecutive observations at relative time points, $T \subseteq \mathbb{N}$. Formally, each x_t is an m-dimensional vector, $x_t \in \mathbb{R}^m$, where x_t^k refers to the value of the kth variable of x_t for $1 \leq k \leq m$.

Consider a black-box method $\mathcal{BB}_{\mathcal{AD}}$ trained based on observations, $O \subset X$, to determine the anomaly status of X at current time point i according to the sequence of observations within a given look-back window $w_i = \langle x_{(i-p)}, ..., x_{(i)} \rangle$ of length p. The status is expressed as a scaled *anomaly score*, α_i, and an *anomaly label*, β_i, assigned to w_i:

$$\mathcal{BB}_{\mathcal{AD}}(w_i) \rightarrow (\alpha_i, \beta_i) \tag{6.1}$$

$\mathcal{BB}_{\mathcal{AD}}$ calculates the anomaly status on the basis of its internal logic learned from the training set associated with X. Therefore, the model's input contains causes as the particular feature configurations for this instance that the model used for prediction. These causal relationships between the

assigned status (α_i, β_i) and the related observations from w_i are to be explained as *interpretable characteristics* in order to make the black-box intelligible. Such interpretable characteristics should enable a comprehensible causal chain of reasoning—for example, as in "having a spike bigger than the threshold isolates this observation from the expectation".

Let f_i be the set of all possible causal characteristics that can assign (α_i, β_i) to w_i. Then, the goal is to identify the most significant, conspicuous, and ideally intuitive causal characteristics f_i^*, $f_i^* \subseteq f_i$, associated with Expression 6.1. Specifically, if the distinguishing causal characteristics (or their underlying patterns) disappear from the input, the black-box's result may change significantly. This goal is referred to as the **Temporal Anomaly Detection Explainer (TADExp)**. Thus, given w_i as input, ${F_i}^*$ is supposed to be provided as follows:

$$TADExp(W_i, \mathcal{BB}_{\mathcal{AD}}) \rightarrow {F_i}^* \tag{6.2}$$

Furthermore, the model's local reasoning may differ from one input w_i to another w_j owing to various significant latent patterns in local spaces (i.e., ${f_i}^*$ may differ from ${f_j}^*$ for $j \neq i$).

6.3.2 Addressed Challenges

The restrictive closed-world assumption [284] and other fundamental challenges that anomaly detection concepts face can mislead the model and generate a high false-alarm rate [260]. Moreover, the vulnerability of advanced anomaly detection techniques to adversarial scenarios is a major barrier to their application in critical systems. Well-performing complex models might be easily fooled with very slight or humanly imperceptible perturbations [212, 291]. Therefore, such approaches might not be completely effective or robust against various types of attacks.

To the best of our knowledge, within the literature related to anomaly explanation or description, no model-agnostic anomaly detection XAI for time series has been reported, let alone an explanation of anomalies in multidimensional stream data. Most generic explainability frameworks are based on simplistic presumptions such as independent and identically distributed random variables, balanced datasets, linearity, and supervised settings, which are unrepresentative of the challenging nature of anomaly detection in stream data due to nonlinear and nonstationary processes, latent confounding causes, potential noise, etc. Moreover, anomaly detection explainers are configured for supervised settings and focus on explaining the anomalous observations distinguished by the classifier model. Meanwhile, this is not the case in the context of stream analysis using a semi-supervised model.

6.3.3 Interpretation approach

Theory. The outlying status of isolated events might occur for various reasons; notable anomalous scenarios are as follows:

- Potential short- or long-term anomalies–also called trend breakout or changes–which are significant in the marginal distribution of univariate time series data.

$$P(X^i \in A) \text{ is unlikely}, A \subseteq \mathbb{R}^i$$

- Multivariate changes in feature associations; in certain cases, the marginal distributions remain the same, but the joint feature distribution might differ (e.g., covariance changes).

$$P((X^j, ..., X^k) \in A) \text{ is unlikely}, A \subseteq \mathbb{R}^{j..k}$$

- Multivariate changes of temporal aspect; some anomalies may be latent in the tails of the time dimension, for example, outside the temporal context or out-of-order events.

$$P((X^j_{i..l}, ..., X^k_{i..l}) \in A) \text{ is unlikely}, A \subseteq \mathbb{R}^{j..k} \text{ and } \{i..l\} \in T$$

Some of the known temporal change scenarios may cause a variety of anomalies. For example, *trend mean-shift* occurs because of sudden process changes. *Trend ramp up/down* is a gradual transition process from one stable state to another compared to the mean-shift. *Pulses* are mainly related to specific and sharp changes such as the predictor state transitioning (i.e., increasing or decreasing) from its typical state for a while and then revert to the normal status after a certain interval; this break can last from a very short peak/valley to a wider sinusoidal or stage peak.

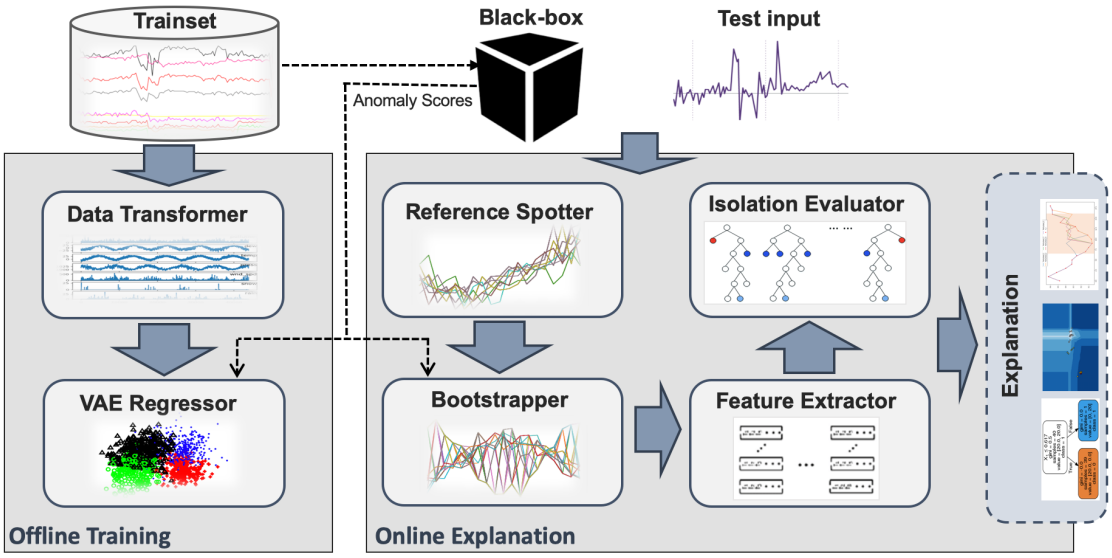


Figure 6.1: *TADExp* Framework

Explanation definition

The main goal of explaining an anomaly detection technique, $\mathcal{BB}_{\mathcal{AD}}$, is to highlight the reasons behind its *anomaly/normal* claims about the given input w_i. An anomalous input stands out from the normal cases according to its isolating characteristics, but the normal events should only be justified based on their normality in comparison to the other normal events. Thus, a semi-supervised explanation can be formally defined based on the $\mathcal{BB}_{\mathcal{AD}}$'s normal expectation in the given temporal context, $\{w'_i \mid \beta'_i = Normal\}$. To this end, an intuitive and interpretable representation can be configured as follows:

- **Difference analysis.** To delineate why an input w_i, is anomalous ($\beta_i = Anomaly$), a user needs to know how it could be converted into a normal case in its context ζ_i with minimum cost, $w_i \xRightarrow{\text{Min}_{\text{dist}}} w'_i$. Ideally, the corresponding normal observation(s) $w'_i \in W'_i$, which represents the model's expectations, should have the minimum distance relative to the given input.

- **Similarity analysis.** To delineate why an input w_i is normal ($\beta_i = Normal$), the user expects to see patterns similar to w_i in the previously observed cases w'_i by the model.

Therefore, w'_i should have the following property:

$$\{w'_i \mid \mathcal{BB}_{\mathcal{AD}}(w'_i) \rightarrow (\alpha'_i,\ Normal)\ \&\ w_i \xRightarrow{\text{Min}_{\text{dist}}} w'_i\} \tag{6.3}$$

Interpretable data representations

It is important to distinguish between the model's input and interpretable data representations to explain a black-box model. An interpretable explanation must employ a humanly comprehensible representation regardless of the actual features used by the model.

Analyzing the raw time series is computationally expensive and unintuitive in terms of features contributing to the task of interest. However, deeper-encoded theoretical concepts and measures derived from the shapes of temporal sub-sequences can enhance the quality of provided explanations and extend our understanding of sequential and temporal patterns. For example, we might learn that suspicious cyberattacks have signal intervals with lower entropy, or include some fluctuations following the onset of a trend ramp-up.

Therefore, *TADExp* utilizes a *temporal feature generator (TFG)* component to abbreviate each input window, w_i, into a set of artificial measures ($\omega_i = \text{TFG}(w_i)$) as its unique signature by capturing both "local" and "global" characteristics. After thoroughly reviewing the literature [58, 73, 97, 230, 231, 277] on time series quantitative statistical features, we utilize the following measures as the core engine to quantify a wide range of time-series properties: serial correlation, skewness, kurtosis, mean, median, standard deviation, chaos, energy, entropy or majority (for categorical features), periodicity, number of peaks, minimum, and maximum. Moreover, the same type of artificial measures is generated based on highly associated features. This measure set is extendable or replaceable with more complex functions [58] according to the user's interest. However, the user

should be aware of each representation's benefit in terms of the type of understanding that it can provide about a given problem.

Next, given the feature correlation and prediction strength analysis, the final set of measures will be summarized on the basis of multiple test procedures. Some advantages of this approach are that it (1) captures most essential information independent of the window size, (2) is relatively insensitive to potential noise or missing values, and (3) is highly intuitive because of the type of the extracted interpretable features. It is worth noting that any choice of interpretable representations has some limitations. Especially in presence of complex $\mathcal{BB}_{\mathcal{AD}}$, the generated temporal features may not be sufficiently advanced to truly represent the black-box's behaviors.

Fidelity-interpretability trade-off

As mentioned earlier, *TADExp* acts over a set of interpretable characteristics, $\mathcal{F}$, extracted from the original temporal inputs $\mathcal{W}$. Since not every combination of $\mathcal{F}$ and *TADExp* may be simple enough to be understandable by humans, we clarify the existing fidelity–interpretability trade-off by following the same notation as previously reported [234]. In the following, we omitted index i for convenience. Let us consider π_w as a proximity measure between w and the instances in its local cluster $w' \in C(w)$.

To make the explanation faithful, the loss of *TADExp* should be minimized in approximating the causal reasoning of $\mathcal{BB}_{\mathcal{AD}}$ in the locality of w ($\mathcal{L}(TADExp, \mathcal{BB}_{\mathcal{AD}}, \pi_w)$). Meanwhile, the complexity of the representative causal characteristics provided by *TADExp* should be minimized in order to be interpretable to humans $\Omega(TADExp, f)$. Thus, the provided explanation is defined according to the following trade-off:

$$\xi(w) = \underset{TADExp, f}{arg\ min}\ \mathcal{L}(TADExp, \mathcal{BB}_{\mathcal{AD}}, \pi_w) + \Omega(TADExp, f) \tag{6.4}$$

Various *TADExp* techniques, proximity measures π_w, complexity measures $\Omega((TADExp), f)$, and fidelity functions $\mathcal{L}$ can be replaced in this formula. However, in this present study, we configure *TADExp* based on a hybrid of case base reasoning, CBR, and counterfactual explanation, CE, clustering-based weighted techniques to indicate the local proximity π_w and a set of statistical and structural characteristics to realize $\Omega((TADExp), f)$.

Clustering for local exploration

In order to have a model-agnostic explainer, the loss of $\mathcal{L}(\text{TADExp}, \mathcal{BB}_{\mathcal{AD}}, \pi_w)$, considering its local neighborhood, must be minimized. However, sample perturbation is very tricky and can cause various anomalies in temporal anomaly detection configurations. Therefore, to realize π_w, a set of weighted reference samples $\mathcal{W}' \approx \mathcal{W}_{\mathcal{R}}$ is drawn from the historical observations most similar to w. To this end, the train set should be clustered to put similar cases in the same bin and find their representatives for the explanation phase $D \xrightarrow{Cluster} \{\zeta_1, \zeta_2, ..., \zeta_K\}$.

However, to maintain a reasonable fidelity to $\mathcal{BB}_{\mathcal{AD}}$, it is preferable to base observation clustering on associated anomaly scores. This comparison method reflects the observations' similarity from the black-box's perspective.

$$\zeta_k = (\mathcal{W}_k, \mathcal{A}_k) = \{(w_j, \alpha_j)\} | \mathcal{W}_k \subset \text{segment}(\mathcal{O})\} \tag{6.5}$$

In general, this process can be realized with a regression clustering method that considers a family of limited functions $\Phi = reg$ and a loss function ($e() \geq 0$) to solve the following minimization problem:

$$\text{reg}^{\text{opt}} = \underset{\text{reg} \in \Phi}{arg\ min} \sum_{i=1}^{N} e(\text{reg}(\text{TFG}(w_i)), \alpha_i) \tag{6.6}$$

While there are various approaches to approximate this regression clustering optimization, here we utilize a variational autoencoder (VAE)-based regression model [77, 295]. Thus, the cluster $C(w)$ of a test observation w can be dynamically found according to the similarity of its embedded representation in latent space in comparison with the embedded train set observations, $\mathcal{O}$.

$$C(w) = \{w'_j | \text{encoder}(w'_j) \in \text{K closest neighbors of encoder}(w)\}$$

As VAEs are unsupervised generative techniques, they are expected to tackle the latent space irregularity problem and model the distribution of each cluster of training data in the compressed manifold of their latent space [77]. A VAE consists of a probabilistic encoder ENC : $\mathbb{R}^m \rightarrow \mathbb{R}^d$ and a decoder DEC : $\mathbb{R}^d \rightarrow \mathbb{R}^m$. The encoder defines a distribution over latent codes $q(z_i|w_i)$, typically in a two-step process of mapping $w_i \rightarrow z_i(\mu_i, \sigma_i)$ followed by sampling z_i from a Gaussian distribution with these parameters. Next, it decodes the samples z_i into a mean value and standard deviation of the output variable and finally samples from the output variables distribution to reconstruct the inputs $p(\hat{w}_i|z_i)$.

However, in VAE-based regression, the VAE is expected to be associated with each w_i with a latent representation $z \in \mathbb{R}^m$, which is dependent on its anomaly score α_i. Thus, the modeling of latent representations differs from the traditional VAE. In this structure, instead of using a single Gaussian prior, z is explicitly conditioned on an anomaly score α, such that the conditional distribution $p(z|\alpha)$ captures an anomaly score specific prior to latent representations. $p(z|\alpha)$ can also be called a latent generator, from which latent representations can be sampled for a given anomaly score.

The parameters of this generative model are optimized by maximum likelihood estimation–that is, by maximizing the sum of log likelihood $\sum_{i=1}^{N} log(p(w_i))$. However, the standard procedure of variational inference is modified by introducing an auxiliary function $q(z_i, \alpha_i|w_i)$ to approximate the true posterior $p(z_i, \alpha_i|w_i)$ [295]. Thus, in addition to probabilistic encoder $q(z_i|w_i)$, the model includes one probabilistic regressor $q(\alpha_i|w_i)$. Hence, the latent representations are linked with the predicted anomaly scores, because (1) latent representations generated from the predicted α must

resemble the latent representation of the input w, and (2) score-linked variation in the latent space is encouraged to follow a direction associated with α.

Now, given this set of validated reference cases $C(w) = \mathcal{W}_{\mathcal{R}}$, and their encoded difference with w, π_w is realized, so Equation 6.4 can be approximated using a systematic explainer to obtain the desired explanation, $\xi(w)$.

Hybrid explanation

Here, we describe a concrete instance of the proposed generic framework that addresses the challenges discussed above by comparing the anomalous input w_i with the collected reference cases $\mathcal{W}_{\mathcal{R},i}$ and highlighting the main characteristics that contribute to its isolation. This instance also presents the most similar entities among reference cases $\mathcal{W}_{\mathcal{R},i}$ to explain why a given input is normal.

As formulated in Algorithms 1 and 2, the explanation process includes one offline training stage that will be executed only once to prepare the explanation platform as well as another online stage for explaining the input. First, the required transformations are applied to handle non-stationarity of the normal train set. After segmenting the train set along with its corresponding anomaly score signals, the VAE-based regressor is fitted to model the latent space of the train set and the corresponding anomaly scores per Section 6.3.3. In the online stage, as Algorithm 2 describes, the test window, w_i, will be transformed by passing through the same prepossessing steps. Then, its most relevant cases $\mathcal{W}_{\mathcal{R},i}$ are found by utilizing the dynamic time warping (DWT) similarity measure [251].

The final inference about the model's reasoning is drawn on the basis of the test input and its own reference cases $\mathcal{W}_{\mathcal{R},i}$ from the two main perspectives of *similarity evaluation* and *isolation evaluation*: the former asks, "how close are the most similar normal cases to the given input?" ($\text{Sim}(w_i, \mathcal{W}_{\mathcal{R},i})$) and the latter asks, "how different is the given input from the normal cases in its own context?, and how significant are its differences?" ($\text{Diff}(w_i, \mathcal{W}_{\mathcal{R},i})$). These results will ultimately be explained and illustrated using interpretable plots.

Isolation evaluation. The main goal of isolation evaluation is to look up the characteristics that contribute to distinguishing this window from the reference cases $\mathcal{W}_{\mathcal{R},i}$ and is realized through a counterfactual explanation strategy in *TADExp*. The overall procedure is as follows:

1. Test window upsampling is applied using *step-wise block bootstrapping(SWBB)* to generate slightly different versions of the test window $\mathcal{W}_{\mathcal{B},i}$ labeled as anomalous by the black-box model (see Figure 6.2).

2. By utilizing TFG component, the interpretable characteristics for $D' = \{w_i \cup \mathcal{W}_{\mathcal{R},i} \cup \mathcal{W}_{\mathcal{B}',i}\}$ are generated and subsequently filtered according to the most relevant characteristics.

3. Interpretable models of iforest are trained based on D', and the distinguishing features that contribute to the black-box's decision about w_i are extracted; iforest is iteratively retrained in the absence of the extracted features and their highly correlated and confounded features.

4. The extracted causal features are sorted by training a decision tree regressor based on the pruned dataset D' in the previous phase; thus, the order of contributing characteristics are found based on their correlation with the anomaly scores.

5. Finally, the anomalous sub-intervals are found; if there is a commonality in the replaced intervals in the subset of bootstrapped windows, which are categorized to the same branch as w_i, it is highlighted as an anomaly center; any pattern-matching algorithm can be used in this step.

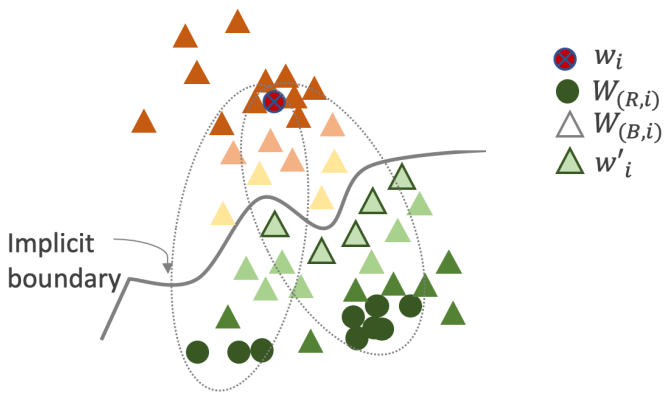


Figure 6.2: A representation of upsampling using step-wise bootstraping

Step-wise block bootstrapping. Simple re-sampling does not work for time series anomaly detection because it destroys the dependence between successive values in the time series [222]. Thus, inspired by block bootstrapping [222], we have devised an effective step-wise algorithm that works and surrogate data with the same sort of dependence as the original latent space of cluster. As described in Algorithm 3, in brief, it swaps random blocks of the full test window w_i with the same size and position in reference windows $\mathcal{W}_{\mathcal{R},i}$. These sorts of replacements generate bootstrapped cases $\mathcal{W}_{\mathcal{B},i}$ consisting of random blocks from both reference and test window cases w_i. In the next step, $\mathcal{W}_{\mathcal{B}',i}$ is generated by filtering and extracting a subset of anomalous perturbed windows whose embedded representations are close to both $\mathcal{W}_{\mathcal{R},i}$ and w_i.

Algorithm 1: Offline Training

Input: $(O_{train}, \mathcal{BB}_{\mathcal{AD}})$: (train set, temporal anomaly detection black-box)
Output: (encoder, $\mathcal{Z}_{\text{latent}}$): (VAE encoder, embedded train set)

```
O'_train ← handleNonStationarity(O_train)
O_seg, α_seg ← segmentData(O'_train, BB_AD(O'_train))
VAEregressor = (encoder, decoder)
train(VAEregressor, (O_seg, α_seg))
Z_latent = encoder(O_seg)
return encoder, Z_latent
```

Algorithm 2: Online Explanation

Input: $(w_i, \mathcal{Z}_{\text{latent}}, \mathcal{BB}_{\mathcal{AD}})$: (anomalous input, embedded train sets, black-box model)
Output: F_i^*: sorted causal features

$w_i \leftarrow \text{transform}(w_i)$
$\mathcal{W}_{(\mathcal{R},i)} \leftarrow \text{NN}_{\text{DWT}}(\text{encoder}(w_i), \mathcal{Z}_{\text{latent}})$
$\mathcal{W}_{(\mathcal{B},i)} \leftarrow \text{SWBB}(w_i, \mathcal{W}_{(\mathcal{R},i)})$
$D' \leftarrow \text{TFG}(\{w_i\} \cup \mathcal{W}_{(\mathcal{R},i)} \cup \mathcal{W}_{(\mathcal{B},i)})$
$D' \leftarrow \text{selectFeatures.transform}(D')$
$\mathcal{F}_i^* \leftarrow \{\}$
Loop
  $est \leftarrow \text{iforest}(D')$
  $\gamma \leftarrow \text{extractIsolatingFeatures}(est)$
  if *contb is not isolating* **then**
    break
  $\mathcal{F}_i^*.\text{add}(\gamma)$
  $D' \leftarrow \text{removeFeatures}(D', \gamma)$
  $D' \leftarrow \text{pruneClusters}(D')$
$\text{DT} \leftarrow \text{decisionTreeRegressor}(D'[\mathcal{F}_i^*])$
$\mathcal{F}_i^* \leftarrow \text{getSortedFeatures}(\text{DT})$
$\Lambda \leftarrow \text{findAnomalousIntervals}(\text{DT}, w_i, \mathcal{W}_{(\mathcal{B},i)})$
return $\mathcal{F}_i^*, \Lambda$

Algorithm 3: Step-wise Block Bootstrapping (SWBB)

Input: $(w_i, \mathcal{W}_{(\mathcal{R},i)}, \mathcal{BB}_{\mathcal{AD}})$: (anomalous input, Reference train cases, black-box model)
Output: $\mathcal{W}_{(\mathcal{B},i)}$: Bootstrapped input

$\mathcal{W}_{\mathcal{B},i} = \{\}$
for *p: 1 to P* **do**
  $q \leftarrow \text{random}(\max : len(w_i)/2)$
  $\text{st} \leftarrow \text{random}()$ & $\text{end} \leftarrow \text{st} + q$
  $W_{\text{sub}} = \{\}$
  $W_{\text{rand}} \leftarrow \text{random.choice}(\mathcal{W}_{(\mathcal{R},i)})$
  for $w_r \in \mathcal{W}_{rand}$ **do**
    $W_{\text{sub}}.\text{add}(w_r[:\text{st}] + w_i[\text{st:end}] + w_r[\text{end}:])$
    $W_{\text{sub}}.\text{add}(w_i[:\text{st}] + w_r[\text{st:end}] + w_i[\text{end}:])$
  $\mathcal{W}_{(\mathcal{B},i)}.\text{append}(W_{\text{sub}})$
$\{\mathbb{A}_{(\mathcal{B},i))}, \mathbb{B}_{(\mathcal{B},i))}\} \leftarrow \mathcal{BB}_{\mathcal{AD}}(\mathcal{W}_{(\mathcal{B},i)})$
$\mathcal{W}_{(\mathcal{B},i)} \leftarrow \{w \in \mathcal{W}_{(\mathcal{B},i)} \mid \beta_j = \text{'anomaly'}\}$
$\mathcal{Z}_{(\mathcal{B},latent)} = encoder(\mathcal{W}_{(\mathcal{B},i)})$, $z_i \leftarrow \text{encoder}(w_i)$
$\mathcal{W}_{(\mathcal{B}',i)} \leftarrow \text{NN}_{\text{DWT}}(z_i, \mathcal{Z}_{(\mathcal{B},latent)}) \mid \text{DWT}(\mathcal{Z}_{(\mathcal{B},latent)}, \mathcal{Z}_{(\mathcal{R},latent)}) \leq \text{DWT}(z_i, \mathcal{Z}_{(\mathcal{R},latent)})$
return $\mathcal{W}_{(\mathcal{B}',i)}$

Similarity evaluation. By taking advantage of case base reasoning, *TADExp* looks for the m closest cases in comparison to the test window from the reference cases $\mathcal{W}_{\mathcal{R},i}$. As a modified instance-based learning technique, our descriptive method goes beyond the generalization of the train set to an abstract model: it aims to explain the underlying nature of the target data on the basis of the previously observed cases. Based on some studies [63, 216], in many different contexts, the users prefer to see very similar real cases instead of general rule-based explanation.

In general, it is expected that a normal test window will be close to some of the samples in its cluster, whereas an anomalous one will be quite different.

$$\begin{aligned} \text{Diff}(w_a, \mathcal{W}_{\mathcal{R},a}) \gg \text{Diff}(w_n, \mathcal{W}_{\mathcal{R},n}) \ \& \\ \text{Sim}(w_a, \mathcal{W}_{\mathcal{R},a}) \ll \text{Sim}(w_n, \mathcal{W}_{\mathcal{R},n}) \end{aligned} \tag{6.7}$$

This approach follows Hawkins idea [128] that, "a sample containing anomalies would show up such characteristics as large gaps between *outlying* and *inlying* observations and the deviation between outliers and the group of inliers, as measured on some suitable and standardized scale." If the contrary happens, the provided explanation might contribute to the model assessment and provide the required insight to improve it.

6.3.4 Experimental Evaluation

In our experimental evaluation, we need to answer the following questions corresponding to the claims made in Section 6.1:

1. What is *TADExp*'s performance in discovering the features contributing to the BB model's decision? To this end, *TADExp*'s average performance is measured in identifying the significant characteristics.

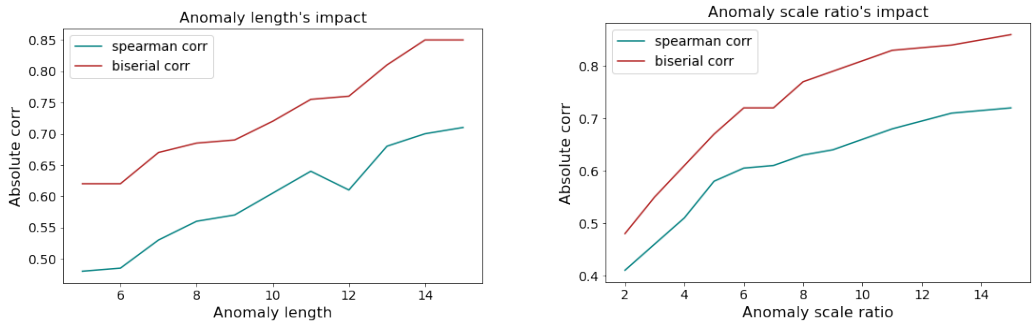
2. Is *TADExp* able to detect complex causal patterns behind events? To answer this question, we measure how the explanation quality changes as the anomalies' scale ratio or length of the collective anomalies, i.e. the number of temporal steps involved in the event, are modified.

3. How robust is *TADExp* concerning noisy data? To evaluate this property, we assess the method's results in the presence of different noisy features embedded in the train set.

4. How strong is the explanation process? The internal validity of *TADExp* can be influenced by both search performance in finding the reference cases $W_{(R,i)}$ and consequently finding representative characteristics and their fidelity to the black-box model's internal logic. Evaluating the impact of detected factors by *TADExp* on the model's results can address the second property to some extent. Also, a quality analysis of reference cases found by the clustering stage targets the former.

Assumptions. *TADExp* holds the following assumptions regarding the input stream data:

- The given time series represents a causal process; otherwise, it would not apply to real-time systems and our framework would not handle it.



(a) (b)

Figure 6.3: Average absolute correlation of the provided explanation with the BB's results with respect to a) the length of anomalous patterns (l), b) the injected anomaly scale ratio (r).

- The underlying time series is regular, or almost regular; otherwise, it should be stationary.
- The training repository is updated systematically if the data is evolving and the future observations are very different from the previous ones.
- All the non-stationary time series should be time-aware so they can be decomposed.

Dataset Description

The underlying time series is a synthetic wave which was generated by adding up a few sine waves and some noise. We then utilized a modified version of temporal anomaly generator [148] to add different types of anomalies to the test set of this time series. The anomaly generator simulates extreme, shift, trend, and variance related anomalies. These anomalies can be in different length l, frequencies n, or scales rate r. We set the default values of the length of causal anomalies (l), and scale rate (r) to 10 and 6, respectively; and vary them in Section 6.3.4. In the end, we have trained a semi-supervised LSTM model to find the anomalous cases based on their error range.

Impact evaluation approach. Since there is no specific ground-truth for XAI tasks, one main challenge lies in how to evaluate the quality of *TADExp*'s results and guarantee that the interpretations are indeed faithful to the original model. Some studies perturb the detected features to measure their impact on the model's decisions. However, the detected characteristics by *TADExp* are not the raw features, observed by the model. So, we have devised the following strategy. Utilizing Algorithm 3, we generate a set of normal and anomalous perturbed samples, $\mathcal{W}_{(\mathcal{V},i)}$, which are local to both w_i and $\mathcal{W}_{(\mathcal{R},i)}$; this fills in the gaps between references and the target-of-interest. Then, the explained characteristics' association with the assigned anomaly scores and labels are verified using Spearman's rank coefficients (s_r) and point biserial correlation coefficient (r_{pb}), respectively. We believe a significant coefficient confirms the fidelity of the provided explanation. We perform these analyses for *iter* times to reduce randomness influence.

Robustness to anomalies length and scale rates

TADExp's average performance ($iter = 20$) with various lengths ($5 \leq l \leq 15$) is shown in Figure 6.3.a. Spearman's rank and point biserial correlations of the discovered causal characteristics are relatively stable around 48% and 62% concerning very short profile lengths, where the differentiation of the pattern offset is more challenging. However, these measures slightly increase to above 71% and 85% for longer anomalies. Also, the same analyses concerning various scale rates ($2 \leq r \leq 15$) is shown in Figure 6.3.b. The association of the discovered causal characteristics with the model's results is relatively stable around 72% and 86% with respect to larger scale-rates, but it sharply falls off to above 41% and 48% for smaller scale-rates, where the anomalous cases are closer to the normal observations. The results show that *TADExp*'s performance remains acceptable even for complex anomalous patterns like a very short or limited range of scales.

Robustness to noisy datasets and features

To this end, we change the settings of the original data generator to create a multivariate synthetic dataset. But, we only embed the anomalies in the main feature $\{f_1\}$, and add some white noise to the other two (irrelevant) features $\{f_2, f_3\}$. Then, we train three different semi-supervised anomaly detectors using LSTM, autoencoders (AE), and support vector regression (SVR). Based on our experiments, as Figure 6.4 illustrates, the percentage of times that artificial characteristics generated based on irrelevant features appear in *TADExp*'s explanations is almost negligible.

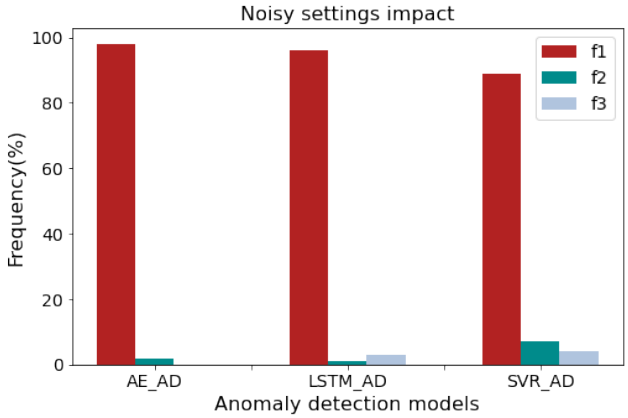


Figure 6.4: TADExp's robustness in presence of noisy features

Quality of local neighborhood

To have a high-quality explanation, we also need to make sure of locality of the detected reference cases and perturbed samples (considering the BB's reflection). Let's consider the generated test set shown in Figure 6.5 along with the predicted anomaly scores and status by an LSTM-based anomaly detector (LSTM_AD). For this specific experiment, *TADExp*'s VAE achieves a mean squared error of 0.006 in predicting the anomalous samples, which is reasonable for values between 0 and 1.

Moreover, the average error of 0.097 and 0.113 for reconstructing normal validation and test sets, respectively, confirm VAE's quality of the latent representation. The average distance of the $k = 30$ closest reference cases found by $\mathrm{NN}_{\mathrm{DWT}}$ for all the normal test cases in the latent space is 0.12, while this value increases to 1.734 for anomalous test cases. All these statistics indicates our trained VAE's capabilities in embedding similar observations to the same neighborhood.

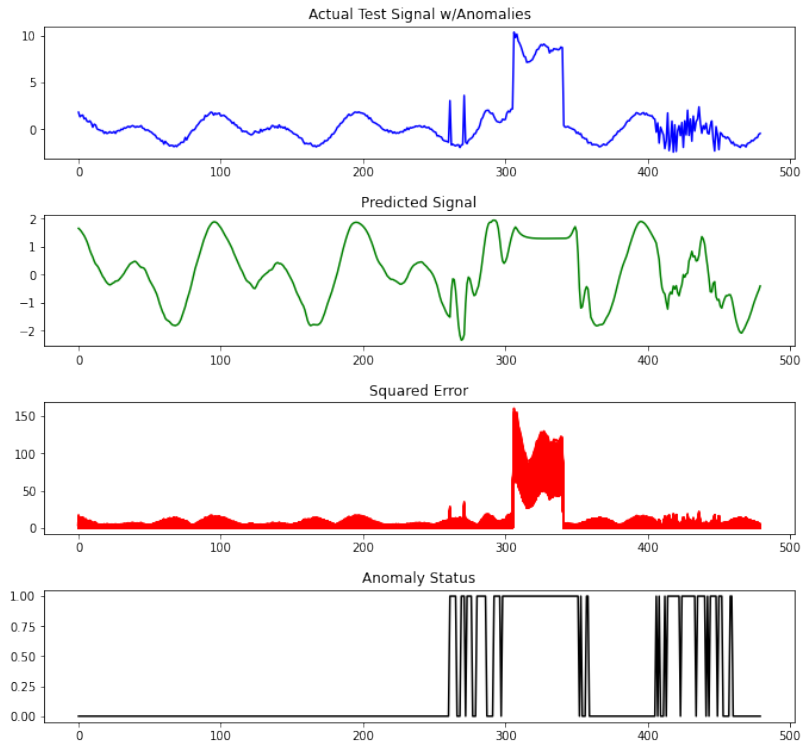


Figure 6.5: a) The actual test signals, b) Predicted signals by LSTM model, c) Prediction squared error, d) The final predicted anomaly status

Running example

Figure 6.7 illustrates a running example of the isolation evaluation process. Let's consider the temporal interval shown in Figure 6.7.a as the target instance (T). Figure 6.7.b shows the transformed representation of T encoded by the trained VAE, in blue, and a few of the nearest transformed reference samples (r) in this latent space. The original representation of the discovered reference samples is shown in Figure 6.7.c. As you see, *TADExp* has not observed any similar case to T, so it matches T with observations from two different scenarios: 1) The normal cases that start from the same maximum pick and continue the decremental trend toward the minimum value in sinusoidal wave, 2) The opposite normal cases that have more commonality with the second half of the target instance, but their incremental trend starts from smaller values.

Then, applying Algorithm 2 generates the perturbed instances illustrated in Figure 6.7.d. The t-SNE representation of these samples in Figure 6.7.e confirms that they keep their locality around our

target of interest (T) and fill the gap between this sample and the normal reference cases ($\mathcal{W}_{\mathcal{R}}$). Figure 6.7.f depicts a contour representation of the target input in presence of the normal reference cases and similar perturbed anomalous cases. This plot provides some sense of the implicit boundary which isolates the anomalous input and the degree to which it is different from the normal observations. The tree-based representation in Figure 6.7.g illustrates the same explanation from a rule-based perspective. Also, Figure 6.7.h represents a few of the approximated counterfactuals based on the most similar normal perturbed samples. Also, this contributes to specifying the main parts of the target instance that cause abnormality status and need to be replaced to change the assigned label. As this case study shows, such an analogy-based explanation is coming from common sense, its reasoning is based on real-world and tangible evidence and highly related to the specific problem domain.

To validate the reference cases discovered by *TADExp*, we extracted a second set of normal cases utilizing the temporal context of target instance (T) (i.e. considering the seasonality and trend), which match the normal trend that the anomalous T was expected to continue (see Figure 6.6). As we see, some of the normal reference cases found by *TADExp* (in Figure 6.7) are very similar to the temporal reference cases; this also demonstrates the quality of local sample discovery process. So the approximated counterfactuals highlight the divergence from real-world expectation, which can be especially helpful in detecting the set of adversarial attacks.

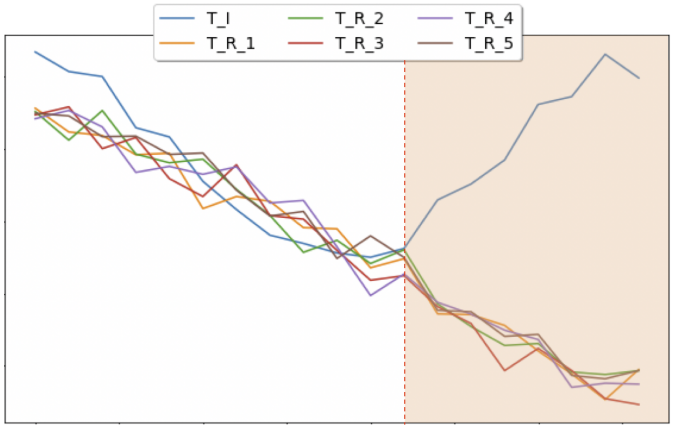


Figure 6.6: Temporal reference cases of T

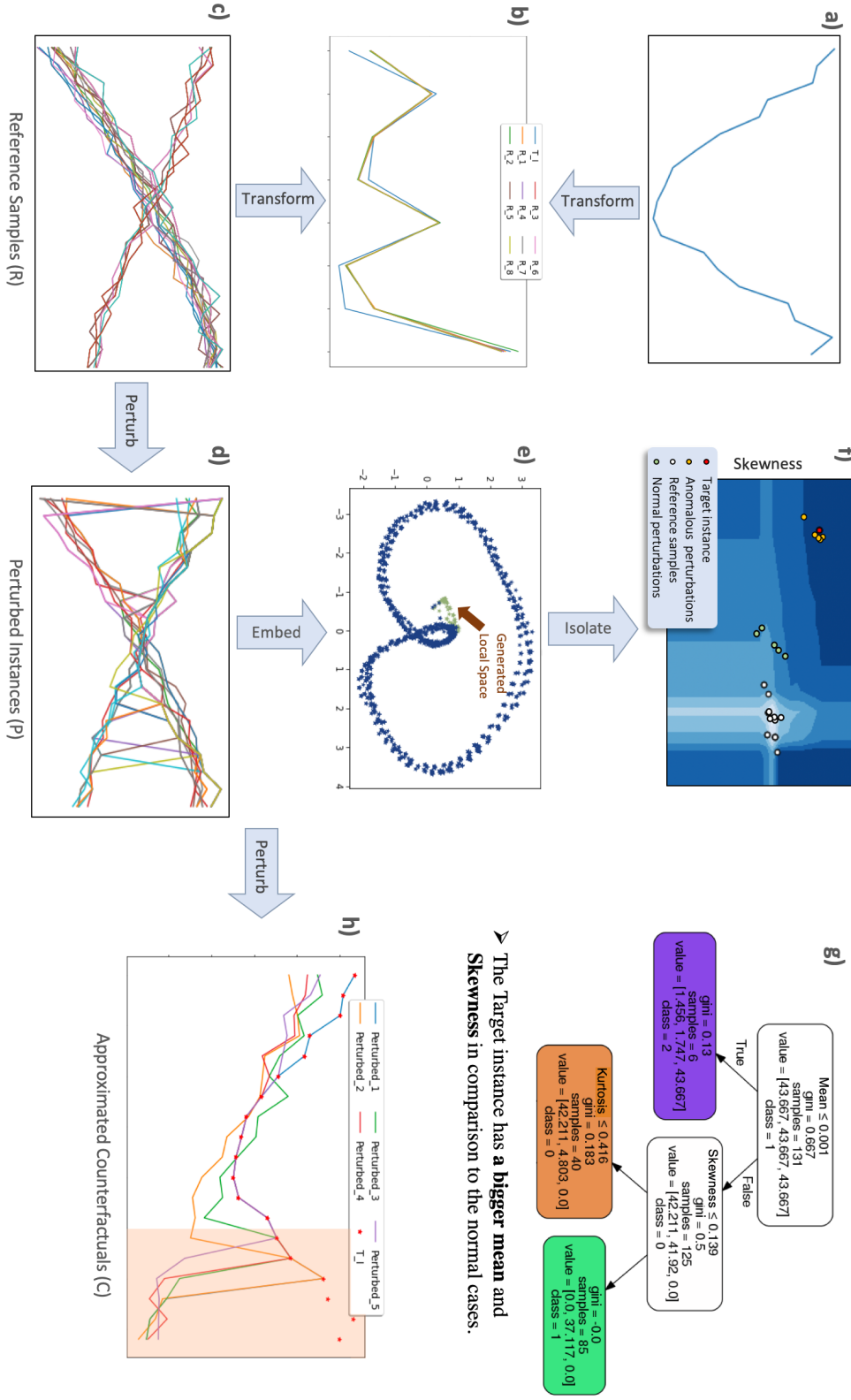


Figure 6.7: A running example of the isolation evaluation process in *TADExp*

6.4 An Interpretable Variational Autoencoder to Find Cyberattacks

Motivation. This section analyzes the existing challenges in the CPS context based on a real-world scenario. Thus, we aim to present an interpretable autoencoder model to distinguish attacks from natural events and analyze the events to find the main source of attacks and their characteristics in order to address the vulnerabilities and prevent future incidents. Autoencoders are specific types of unsupervised ANNs for efficiently learning embedded features of data. An autoencoder basically represents the normal data in lower dimensionality and then reconstructs the data in the original space. One of the main applications of autoencoders is toward learning a more-efficient representation of data for various purposes–for example, dimensionality reduction [155]. Recently, autoencoders have been widely adopted for unsupervised anomaly detection tasks [84, 243]. The overall process of system modeling and anomaly detection is founded based on embedding data and accurately reconstructing them. The autoencoder will generate larger reconstruction errors for different and unobserved inputs, thereby distinguishing anomalies.

Although autoencoder models are very efficient and popular in anomaly detection, they are far from being interpretable. In brief, such models decide based on the whole input and are unable to clarify the discovered patterns, determine the exact time point of attack, or determine the features that contributed to the detected anomalies. Furthermore, to our knowledge, existing XAI methods are mainly proposed to explain a supervised predictor and there are only very limited studies regarding explaining unsupervised anomaly detection methods and they are typically model-specific or need to have access to the model's internal stages [227, 250]. For example, one line of such studies is about techniques that try to explain the latent space by employing a particular interpretable model [162]. Also, combining graphical models with AEs to interpret unsupervised techniques [144, 259] looks promising.

The main challenge of explaining unsupervised trained models, like autoencoders is how to find the contributing features, especially because there is no specific boundary to surrogate it. Thus, we have developed a fast method to distinguish anomalous events representative of cyberattacks and an unsupervised autoencoder model. Our experiments with real-world data confirm the performance and applicability of our proposed solution.

6.4.1 Problem Definition

This project aims to propose an advanced, but interpretable unsupervised anomaly detection algorithm to trace suspicious activities in CPSs like the smart grid. The considered system's behavior is monitored over some time interval T, comprising t consecutive time steps, forms a multivariate time series $X_T = \langle x_t : t \in T \rangle, T \subseteq \mathbb{N}$. Intuitively, each element of this time series is a multivariate process with m observed features ($x_t \in \mathbb{R}^m$), meaning $x_i = (x_i^1, x_i^2, ..., x_i^m)$. Thus, our online framework is desired to learn the hidden patterns of the data and detect suspicious cases, and then interpret the results by highlighting the exact point of anomalies.

6.4.2 Method Description

In light of the motivational points developed above and to address the challenges, we now propose our method for CPSs like the smart grid (Figure 6.8). This framework includes two main components: it first learns the latent space of the collected data using VAE and then highlights the exact attack points using combination of reconstruction error-based and perturbation-based analyses.

Algorithm 4: Interpretable Attack Detection: Training

Input: X_{train}
Output: $(AE, threshold)$: Autoencoder and an optimized anomaly score threshold

$encoder, decoder$ = GenerateNetworks()
$AE = (encoder, decoder)$
trainAE(AE, X_{train})
$error_{reconst} = AE(X_{train}) - X_{train}$
Z_{latent} = encoder(X_{train})
$threshold$ = Max$(error_{reconst})$
trainAE() :
  $input$ = generateBatchData(X_{train})
  z_{mean}, z_{var} = encoder$(input)$
  $noise_z$ = sampleNoise()
  $z = z_{mean} + noise_z * z_{var}$
  $output$ = decoder(z)
  $loss$ = computeLoss$(input, output, z_{mean}, z_{var})$
  backwardLoss()
computeLoss() :
  $loss_{reconst}$ = MSE$(input, output) * batch_{size}$
  $loss_{KL} = -1/2 * (1 + z_{var} - z_{mean}^2 - \exp(z_{var}))$
  $loss_{total} = loss_{reconst} + loss_{kl}$
return $AE, threshold$

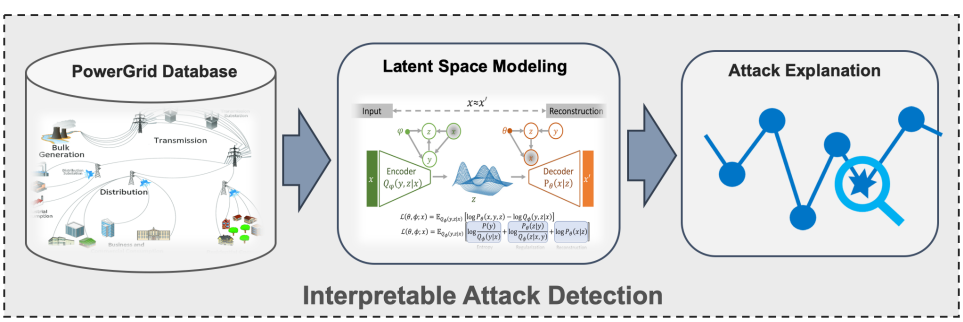


Figure 6.8: Schema of the proposed method

Algorithm 5: Interpretable Attack Detection: Test

Input: $x \in X_{test}, X_{train}, z_{latent}$
Output: $(score_x, label_x, F_k)$: (attack score, status, set of contributive features

```
1  x_reconst = AE(x)
2  score_x = |x_reconst − x|
3  F_k = {}
4  if score_x ≥ threshold then
5      label_x = "suspicious"
6      N_k = topKDivergedFeatures(x_reconst, x)
7      for f_i ∈ N_k do
8          M̂ = findNeighbours(encoder(x), Z_latent)
9          M̂′ = replaceFeature(M̂, f_i)
10         contrib_i = AVG(|AE(M̂′) − M̂′|)
11         F_k.add((f_i, contrib_i))
12 else
13     label_x = "benign"
14 return (score_x, label_x, Sorted(F_k))
```

Latent space modeling

In order to model the compressed manifold of the latent space in the dataset, we train a VAE as shown in Algorithm 4. The VAE model is used because it can tackle the problem of the latent space irregularity [11]. A VAE is an unsupervised generative technique that comprises a probabilistic encoder ENC : $X \rightarrow R^d$ and a decoder DEC : $R^d \rightarrow X$. The encoder defines a distribution over latent codes $q(z|x)$, typically in a two-step process of mapping $x \rightarrow z(\mu, \sigma)$ and then sampling z from a Gaussian distribution with these parameters. Therefore, the VAE encoder returns a distribution over the latent space instead of a single point. Moreover, to ensure a better organization of the latent space, a regularization term is added over the returned distribution in the loss function. This embedded stochasticity helps to obtain perturbed versions of the input.

In this step, we also extract a criterion to determine the suspicion level of a future event. By presuming a normal error distribution for the training observations, the anomaly scores are fitted on the basis of Q-function as the tail distribution function [110, 255]. Thus, $Q(x)$ is the probability that a normal (i.e., Gaussian) random variable obtains a larger value than x standard deviations. Accordingly, the suspicion level of the observations will be determined in comparison to the range of reconstruction errors observed in the training phase.

Attack explanation

In the next step, in order to determine the exact attack point, we propose the procedure shown in Algorithm 5. In unsupervised techniques, the causal reasons behind the detected anomaly are only latent in the input values in comparison to the rest. Thus, the reconstruction error of autoencoders

is a good source of knowledge to explain them. On this basis, first, the features whose reconstructed values differ from the original values are found. To this end, the normal Mahalanobis distance for the features is calculated. Applying a distribution-based distance helps to have an initial ordering of the features that have contributed to the obtained reconstruction error difference.

The next step permutes each candidate feature value to find their range of impact on the obtained anomaly score. For this purpose, the set $\hat{M}$ closest neighbors of the test observation x are found according to their embedded representation in the latent space encoder$(x) \leftrightarrow z_{latent}$. This facilitates a fair distance evaluation independent of unimportant features. Then, the impact of each feature f_i is evaluated by inserting its value in the set of $\hat{M}$ and evaluating its impact significance. Because of the features dependencies, two different scenarios might occur: (1) a *contributive feature* in which the feature positively affects the predicted anomalous status; (2) *compensator feature* in which the feature negatively affects the predicted status (i.e., the feature reconstruction difference is caused by feature relations).

6.4.3 Experimental results and Evaluations

This section evaluates the anomaly detection performance and explanation quality of our proposed method.

Experimental settings. The utilized dataset in this study is an open-source smart grid attack repository [130] including both Information Technology (IT) and Operational Technology (OT) attributes [218]. We refer the reader to [130, 218] for further details about its architecture. This dataset includes 128 features and its numerical feature values have been standard scaled. In our experiments, the encoder and decoder are both single-hidden layer networks with 80 dimensions; the latent dimension includes 40 nodes. We used fully-connected NNs in both the encoder and the decoder with *tanh* activation functions. The model parameters are optimized using RMSE loss value through stochastic gradient descent and ADAM step updates. Further, the optimization process is done by a grid search; so potentially there is room for improvement regarding the behavior prediction results.

Peer methods. A few studies have used this dataset in their experiments [71, 218, 219]. However, they have mostly utilized supervised learning to detect the attacks. While our research is focused on unsupervised modeling of the system to detect zero-day attacks. Anyhow, we compare the performance of our method with some of the high-performance supervised and other unsupervised techniques, as follows:

- XGBoost (XGB). Effective implementation of gradient tree boosting.
- Hybrid. A combination of multiple diverse models of Random forest, Extra trees, and K-neighbors optimized by SuperLearner framework embedded in the Mlens package.
- Multi-layer perceptron (MLP). A feed-forward neural network.
- iforest. A tree-based anomaly detection algorithm.

- One-class SVM (OSVM). A modification of SVM to support one-class classification.

Performance evaluation

This section evaluates the performance of our proposed method in comparison with the peer supervised/unsupervised methods. As Table 6.1 shows, our interpretable VAE outperforms the other unsupervised baselines by achieving a higher ROC and F1-score in the presence of a very acceptable recall and precision tradeoff. Also, the performance of our unsupervised VAE is almost comparable with the optimized supervised hybrid method. This shows the capabilities of this technique in learning the latent space of normal data to reconstruct them effectively.

Table 6.1: The performance evaluation of our method

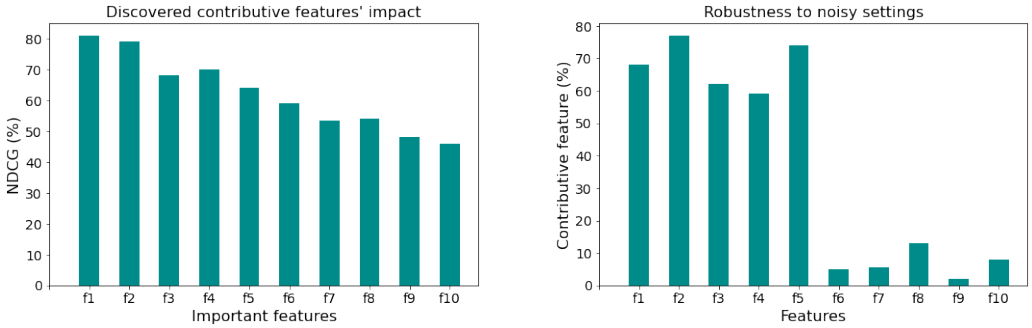
Model	**Precision**	**Recall**	**F-measure**	**ROC**	**Unsupervised**
XGB	0.92	0.96	0.939	0.951	✗
Hybrid	**0.986**	**0.964**	**0.975**	**0.983**	✗
MLP	0.93	0.95	0.969	0.98	✗
iforest	0.763	0.851	0.804	0.818	✓
OSVM	0.798	0.742	0.769	0.75	✓
VAE	**0.932**	**0.957**	**0.944**	**0.966**	✓

Interpretation examples

Result's explanation, especially in the context of anomaly detection, is very new and different in various tasks and domains. Thus, there is no standard evaluation technique or measure to verify their soundness. This research evaluates our local intepretability approach through two different perspectives.

1) Interpretation correctness. This experiment is designed to verify the impact of important features' perturbation on the anomaly score. To this end, we have selected a subset of ($k = 30$) normal observations in the close proximity to the anomalous test case and perturbed their feature values one by one. Hence, we would be able to find the impact of each feature based on its average changes in the reconstruction error. Since this is a local explanation, we do not expect to observe the same ordering of feature importance for all the anomalous samples. Therefore, we measure the explanation quality and effectiveness using *Normalized Discounted Cumulative Gain (NDCG)* [278]. As Figure 6.9.a illustrates the *NDCG* of the most important feature in the first rank is almost 81% and this value does not drop to less than 46% for the top-10 features, which is a very acceptable rate for our interpretation technique.

2) Interpretation robustness. This experiment is designed to evaluate the robustness of the explanation. To this end, we have created a synthetic dataset and labeled its records to normal/anomaly only based on a subset of features $\{f_1, ..., f_5\}$ and filled out the other (irrelevant) features $\{f_6, ..., f_{10}\}$



(a) (b)

Figure 6.9: a) Interpretation performance for top-10 explained features, b) Interpretations' robustness in presence of noisy features.

with the marginal distributions for each label. Then, we configured a VAE for this dataset and obtained the final feature importance for each sample through the same procedure. Figure 6.9.b illustrates the cumulative feature importance detected by our model for every single sample. As we see, top-5 contributive features are almost stable through all the experiments, and the interpretation process has ignored the noisy features.

6.5 Heidegger: Interpretable Temporal Causal Discovery

Motivation. While machine learning and AI have made impressive progress in predictive and descriptive tasks, establishing causality has remained a challenge [262]. As aforementioned, XAI comprises various techniques which seek to improve the interpretability of AI results. Interpretability is much needed to demonstrate the accuracy and value of the results, and engender user trust.

Multiple attempts have been made to develop XAI, but these are inadequately grounded in theories of explanation and range from non-causal to strongly causal explanations; however, non-causal variants of extra-mathematical, and model explanations appear to be receiving greater attention than causal techniques. Some of these studies hold the view "all scientific explanations are causal explanations" (i.e., causal imperialism) [32], while others assume that anything other than a causal explanation is a rationalization or fictionalization [32, 236] and thereby do not have any explanative value. Within causal explanation framework, all non-causal explanations are attempts to articulate how possibly rather than the ideal standard of how actually [115]. In other words, correlative relationships are important and further our understanding. However, correlation does not imply causation. A dedicated class of methods is needed to address the problem of causal discovery.

Causal discovery strives to identify cause-effect relations between a set of candidate variables and an outcome variable of interest, where any change in a causal variable will result in a change in the outcome [262]. Various methods have been introduced to discover causal relations using observational data [263]. However, most of the existing techniques are limited as they are concerned with causality in a restricted sense and only demonstrate if a treatment can change an outcome or not. Inability to determine causal profiles (i.e., a temporal pattern that results in the most significant change in an outcome variable) greatly limits the interpretability of the results; understanding the minimum duration of an event to be effective, treatment-effect persistence, or the time delay between an treatment and the outcome is critical. While some methods are flexible in terms of defining the hypothesis [31, 48, 258], a user must identify the true hypothesis, possibly resulting in erroneous conclusions or data dredging.

Motivated by these issues, we believe our research in [197] can be modified to open a new perspective to provide causal explanations for temporal AD, illuminating the factors driving an event and the outcomes of counterfactual scenarios. This can significantly contribute to intervention activities (e.g., dementia risk reduction, and/or prevention).

6.5.1 Problem Definition

We assume a given dataset with observations at discrete time points for a set of variables and samples. Among the variables, we distinguish the labels assigned by black-box (BB) corresponding to outcome variable Y, the set of candidate variables X, and the set of potentially confounding variables Z. Formally, $Y_s^t (1 \leq t \leq T, 1 \leq s \leq S)$ is the value of the BB's outcome for sample s in t, where T is the number of time points, and S is the number of samples; $X_{s,f}^t (1 \leq t \leq T, 1 \leq s \leq S, 1 \leq f \leq F)$ is the value of fth candidate variable of sample s in t, where M is the number of candidate variables; and

$Z^t_{s,c}(1 \leq t \leq T, 1 \leq s \leq N, 1 \leq c \leq C)$ is the value of cth confounding variable of sample s in t, where C is the number of confounding variables. An $instance$, referred to as (s, t), is a pair of sample s and time point t.

We define CP as the set of all temporal causal profiles. The goal is to identify causal variables associated with the BB's outcome, denoted as X^*, and their corresponding temporal causal profiles, denoted by CP^*, such that elements in X^* and their corresponding patterns in CP^* provide the most significant causal association with the outcome when controlling for the effect of confounders and the other candidate variables. If a causal variable follows the pattern indicated in its corresponding causal profile, the outcome of BB's model will experience a significant change. We refer to this problem as the **Causal Profile Discovery (CPD)**. Hence, each element of the causal profile set, CP, corresponding to a causal variable, $x \in X$, represents a hypothesis, where the treatment group receives the treatment in the form of the identified causal profile. We define a causal profile, $cp \in CP$, based on the two following components:

1. *Treatment Pattern:* A vector of $\{0, 1\}$ values, denoted by cp_p, where 1 means receiving the treatment while 0 means not receiving the treatment.

2. *Pattern Offset:* A vector of time offsets, denoted by cp_o, where each offset indicates how many time points before the current time the corresponding element of treatment pattern is applied.

Meanwhile, the corresponding control group of the hypothesis is defined based on no-treatment profiles. More specifically, each profile in this group has the same *Pattern Offset* as the causal profile, but its *Treatment Pattern* is a zero vector. For instance the hypothesis that "if x_1 exist at time t and does not exist at time $t - 2$ then y at time t changes" (Figure 6.10) corresponds to causal profile (cp_p : [0 1], cp_o : [2 0]), and control profile (cp_p : [0 0], cp_o : [2 0]).

Required Modifications. The original *HEIDEGGER* framework was focused on solving the causal profile discovery (CPD) problem based on a given dataset and was not associated with black-box explanations. However, we believe it can be easily modified to explain temporal anomaly detection black-boxes. To this end, we need to apply two modifications:

1. Predicted labels by the black-box need to be considered as the output variable. This increases the final explanation's fidelity;

2. A systematic data generator must be utilized to produce the required control profile sets. For each candidate causal profile, *HEIDEGGER* receives the corresponding control profiles from the data generator and labels them using the black-box. Following this, *HEIDEGGER* would be able to evaluate each explanation hypothesis utilizing the randomized-block design component and identify the main cause(s) leading to an anomalous prediction.

6.5.2 Addressed Challenges

Most existing XAI techniques are not able to determine causal profiles (i.e. a temporal pattern that results in the most significant changes in an outcome variable), undermining the interpretability of the results. We propose *HEIDEGGER* [197] to address this problem. *HEIDEGGER* goes beyond a temporal causal discovery technique and is capable of detecting underlying causal relations along with their corresponding causal profiles. Further, *HEIDEGGER* is robustly designed to accommodate noise and detect complex temporal treatment-outcome relations. In the context of XAI, we believe this is one of the earliest attempts that goes beyond rationalization and deals with the issue of temporal anomaly detection models by actually determining causal relations.

6.5.3 Method Description

Our CPD method, *HEIDEGGER*, overcomes the previously mentioned challenges by utilizing an efficient graph search algorithm to find potential causal variables and their corresponding causal profiles. A novel randomized block quasi-experimental design (QED) is applied to evaluate the significance of each hypothesis (i.e., the causal variable and associated causal profile). QED is an offline framework for estimating the causal impact of an intervention without random assignment by manipulating the independent variables.

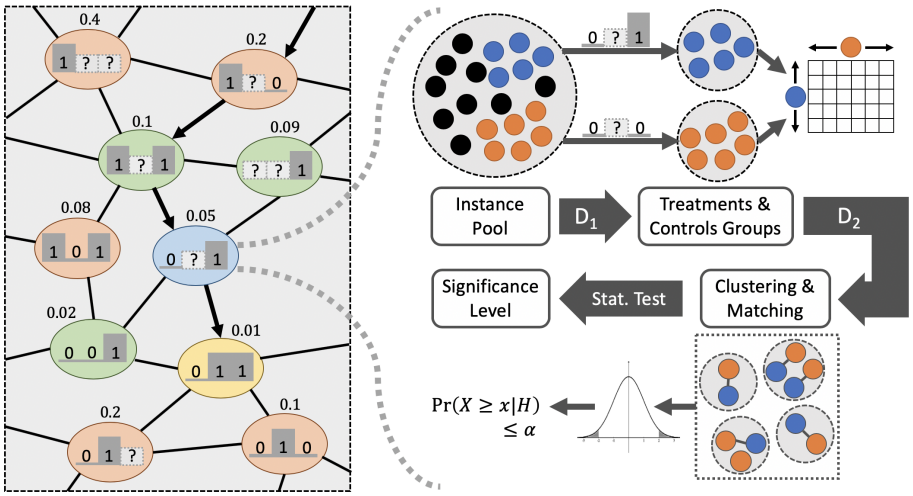


Figure 6.10: *HEIDEGGER* Framework: in each step of the graph search, the most statistically significant of neighbors (green) of the current node (blue) is selected as the current node of the next step. If the current node is more significant than its neighbors (yellow), the run stops. For each node, to compute the significance level, QED process is applied [197].

For each candidate variable the hypothesis search space is modeled as a graph, referred to as the profile graph, where each node indicates a potential causal profile (6.10). *HEIDEGGER* runs a heuristic search for each candidate variable at a time on the profile graph to identify the most significant profile and dynamically generates the local neighborhoods in the search space. Upon visiting each node, the corresponding causal profile is evaluated using a randomized block QED.

This consists of: 1) Identification of treatments and controls; 2) Matching and clustering of instances; 3) Significance evaluation. The significance level of the candidate variables and their causal profiles are adjusted for multiple hypothesis testing based on false discovery rate criteria. A brief description of each phase is presented in the following sections.

Hypothesis search

The strength of *HEIDEGGER* is the traversal of the profile graphs to identify the most significant candidate profile. Each node represents a causal profile, and there is an edge between nodes A and B if the causal profile of node A can be converted to the causal profile of node B with the change of a single element. Given this, the shortest distance between two nodes in the profile graph is equal to their minimum edit distance. The discovery process in Graph is guided by the assumption that the p-values across the network are cohesive, (i.e., hypotheses with similar profiles are expected to have similar p-values). *HEIDEGGER* uses a simple but efficient strategy to build the search space; at every step, it expands the neighborhood of the current hypothesis, and finds the most promising neighboring candidate to be processed by the QED (explained in Section 6.5.3).

To expedite the search, an early pruning mechanism discards the causal profiles with permutation entropy above a given threshold because in practice, the causal profiles are expected to have low permutation entropy. For instance, a high entropy causal profile (cp) such as (cp_p : [1 0 1 1] , cp_o :[8 5 3 1]) is unreasonable, but a causal profile with a lower permutation entropy such as (cp_p : [0 1] , cp_o :[2 0]) is acceptable.

Randomized-Block Design

HEIDEGGER needs to evaluate the significance of the adjacent hypotheses to find the most promising profile in the local neighborhood of a current node, which is likely to lead to the global solution. To this aim, our framework applies QED to find treatment and control groups for each hypothesis and compute the significance of the difference in the outcome between the two groups.

The specific QED, used in *HEIDEGGER* is an extended version of the randomized block design that pairs instances based on their history of confounders, treatments, and the outcome, guarantees the effect of these variables is controlled across time. The randomized block design is adpoted for its flexibility, lack of constraining assumptions, and its proven effectiveness in similar studies [15], while its major disadvantage is confounders must be directly controlled resulting in loss of statistical power. However, *HEIDEGGER* compensates for this by considering combinations of time points and samples as *instances*, increasing the number of effective samples by a factor approximately equal to the number of time points. The process of finding and evaluating the treatment and control pairs is done in the following three main steps:

- Identifying the treatment and control groups by comparing the values of the candidate variable of each instance with the causal profile and the no-treatment (control) profile, respectively.

- Clustering the union of treatment and control group instances into blocks, where all instances in a block are similar enough, and randomly matching treatment and control instances within each block.
- Performing a paired statistical test to evaluate whether there exists a statistically significant difference between the outcome of the two paired groups.

Multiple Comparison Adjustment

Evaluating a wide range of causal profiles for several candidate variables results in a large number of hypothesis tests. Considering that each hypothesis has a specified type (I) error probability, evaluating the set of hypotheses as a whole sharply increases the chance of some type (I) error. Conversely, evaluating the causality of multiple candidate variables and causal profiles for an outcome variable produces a large number of hypotheses [27]. A standard approach to counteract the multiple comparisons problem is to adjust p-values and the threshold to accept them based on the false discovery rate. This is the most reasonable multiple hypothesis correction criteria in the context of CPD, because it is not overly conservative for a large number of hypotheses and does not rely on the tuning of the test statistics [27]. *HEIDEGGER* uses the well-known *Benjamini-Hochberg* false discovery rate adjustment method [27] for this purpose.

6.6 Conclusions

High-performing, complex, black-box models are prevalent and inevitable. Despite the outstanding success of complex methods in many areas, there are some essential doubts about their precision, when exposed to adversarial attacks. Especially, due to assigning non-zero weights to the useless features or out of range values, deep learning models are a known target of attackers. Besides, achieving better numerical results without presenting the required intuition for the reasoning behind decisions is not a sufficient and worthy contribution.

Acknowledging that such techniques are not transparent and fully mature and their transition may be timely, it is wise to develop explainable algorithms to clarify the decision-making logic of such complex models, especially in large-scale and critical deployments. We believe our proposed methods can contribute to the field in two ways. Such explanations expose the users to an intuitive presentation of the local decision-making process of temporal anomaly detection concerning the time dimension. Also, it can help the model developer to find the model's vulnerabilities and improve it, which eventually may lead to higher resilience and precision alongside a lower false alarm rate.

Explainability research is a principal area of innovation to support end-users. Building an explainable situational awareness platform that addresses the various suspicious elements is crucial and cannot be overlooked. Our proposed methods can certainly open new perspectives in various contexts like health and cybersecurity.

Chapter 7

Spatiotemporal Situational Awareness

Nowadays, larger volumes of spatiotemporal data are increasingly collected and studied in diverse domains including epidemiology, transportation, and earth sciences [18]. Spatiotemporal data differs from previously analyzed time series because they have both space and time as the ubiquitous aspects of observations. In fact, this introduces another type of sequential dependencies among measurements induced by the spatial dimension, which brings its own additional challenges that need to be dealt with. Having access to the space information provides the possibility of extracting more exciting patterns from the stream data, however, it may increase analytical challenges because of the dynamical essence of space. Thus, the real-world processes in these domains are expected to be inherently spatiotemporal in nature.

Thus, to specify a few of the extra challenges caused by the space dimension, this chapter describes some of the relevant spatiotemporal projects that I have technically contributed to them and have been listed as a co-author in the outcome publications. This topic does not specify any particular phase in the end-to-end pipeline of anomaly detection. But, it is highly relevant in the case of utilizing the end-to-end framework to detect anomalies in spatiotemporal stream data. The remainder of this chapter is organized as follows. Section 7.1 provides a concise review of spatiotemporal anomaly detection techniques. Then, Sections 7.2 and 7.3 provide a summary of two relevant spatiotemporal studies.

7.1 Background

To the best of my knowledge, there are different types of anomaly detection in spatiotemporal datasets which can be distinguished based on their goal. This section provides a summary of some of these techniques.

7.1.1 Spatiotemporal Anomaly Detection

The main goal of this task is finding the spatiotemporal object whose thematic (non-spatial and non-temporal) attributes are significantly different from those of the other objects in its spatial and temporal neighborhoods, as spatiotemporal anomaly (ST-anomaly) [2].

In [143] a set of temporal logic rules have been proposed to capture normal activities. This relation set includes: *before, contains, overlaps, meets, starts, started-by, finishes, finished-by,* and *equals*. The conditional probability of each event Z given Y can be computed based on these relations and the likelihood of Z will be computed based on all of the observed events till that point in time, to get the final anomaly score as $1 - P(Z)$. Some of the other proposed methods in this group are explained in the succeeding parts.

Density-based spatiotemporal anomaly detection

The density-based methods focus on the density of the available data points in the location dimension. For example, the algorithm proposed in [163] includes the following main steps:

- Clusters the data using a modified DBSCAN. It supports the temporal aspect using a traversal tree to find the spatial and temporal neighbors of each object within a given radius
- Compares the new coming values with the average value of all the obtained clusters
- Detects the spatial anomalies by checking their spatial neighbors
- Verifies the temporal neighbors of the spatial anomalies to find the ST-anomalies [2].

Cluster differentiation based spatiotemporal anomaly detection

The algorithm proposed in [57] is founded based on cluster differentiation, which includes the following four main steps:

- Applies classification or clustering to find the regions
- Reduces the spatial resolution of the data by aggregating them
- Compares the result obtained in two different granularity levels. The objects which are found in fine-grained regions, but not in aggregated regions, will be considered as potential spatial anomalies
- Evaluates the temporal neighbors of the suspected anomaly points to find the final set of spatiotemporal anomaly candidates.

7.1.2 Spatiotemporal Anomaly Tracking

This task focuses on tracking solid anomalies based on their volume through time dimension in a specific space dimension. So, if the volume is higher than the average through time dimension, then it is a solid anomaly [2].

Outstretched-based tracker

Outstretch [285] is proposed to track the anomaly movement patterns over the time periods. This method needs two inputs:

1. Top-K spatial anomalies for each year and a variable r,
2. The region stretch, which is the number of grids to 'stretch by' on each side of an anomaly

For all the years, each of the anomalies from the current year is examined to see if it is framed by any of the stretched regions from the previous year. The anomalies that meet this condition are added to the end of the previous years' child list. As a result, all of the possible sequences over all years get stored in the anomaly tree and can be retrieved for analysis.

Spatial center-based tracker

In [187] Wavelet transformation is applied to the input data to extract its hidden distinct patterns. Then, edge detection with a competitive fuzzy classifier is extended to identify the boundary of region anomalies. The center of an anomaly region will be computed based on the fuzzy-weighted average of the longitude and latitude of the boundary locations. In the end, the movement of anomalous regions can be traced by linking the centers of the anomalous regions in consecutive frames.

7.1.3 Trajectory Anomaly Detection

This task mainly aims to find anomaly trajectories, which is more challenging than analyzing static objects in space and time dimensions. Until now, different clustering, classification, or distance-based methods have been proposed which a few of them are explained in the succeeding parts.

Distance-based trajectory anomaly detection

Trajectory anomaly Detection (TRAOD) [174] performs this algorithm in three main phases:

- Two-level partitioning phase. In partitioning process, the distance of the trajectories which include: *perpendicular distance*, *parallel distance* and *angle distance* are computed.
- Detection phase. The main process in this phase is finding outlying trajectory partitions that do not have enough similar neighbors.
- Anomaly labeling. A trajectory is marked as an anomaly if the summation of the outlying partitions of a trajectory meets the length-based threshold of all the partitions.

Direction and density-based trajectory anomaly detection

TOP-EYE is proposed in [102] as a method that computes an anomaly score for each trajectory, continuously, in an accumulating way. This method considers a decay function to address the evolving

computation of the accumulated outlying scores. It discretizes the space into several smaller grids. TOP-EYE considers two types of outlying trajectories:

- Direction-based anomalies. The direction information of trajectories in each grid is turned into a vector of eight values to indicate the probabilities of moving toward each direction.
- Density-based anomalies. The trajectory density of each grid is computed based on the number of trajectories that pass that grid.

Historical similarity-based trajectory anomaly detection

A method to detect temporal anomalies by emphasizing the historical similarity trends between the data points is proposed in [178]. It records the historical similarity values, as temporal neighborhood vector at each road segment, which includes similarity information of each road segment versus the other road segments. The radical changes in these vectors are marked as anomalies. The approach considers some penalties and rewards for each day, based on the following two criteria:

- How similar is the road to the other road segments, *historically*?
- How similar is the road to the other road segments, *instantaneously*?

The final anomaly score is the summation of the obtained rewards and penalties. Because of considering the other road segments in the analysis, this method is robust to potential population shifts.

Motif-based trajectory anomaly detection

A trajectory anomaly detection based on motion-classifiers is proposed in [177] with the following main steps:

- Extracts hidden motifs, as triples of (feature, time, location), from object paths
- Clusters object movement fragments, based on the motif-based generalization
- Classifies objects using classifiers, which can classify very high-dimensional motif feature space, to distinguish anomalous trajectories from normal cases.

7.2 Mining Vessel Trajectories for Illegal Fishing Detection

Motivation. Maritime domain awareness is the effective understanding of activities associated with the maritime environment that could impact upon the security, safety, economy or environment. Identifying threats as early and as distant from shores as possible requires the ongoing analysis of the continuously changing picture of events in the maritime world to assess threat levels and enable early intervention in illicit activities. Illegal, unreported and unregulated (IUU) fishing is one of the most serious threats to the sustainability of world fisheries [93, 94, 214] and a major threat to marine biodiversity, the sustainability and balance of marine ecosystems, to fish populations

worldwide [54], and to global food security as 4.3 billion depend on fisheries for protein [93]. While the extent of IUU fishing taking place is not known in detail, it is estimated that IUU fishing accounts for up to 30 percent of all fishing activities worldwide [92, 94, 289]. Entities that engage in IUU fishing circumvent conservation and management measures, avoid the operational costs associated with sustainable fishing practices, and may derive economic benefit from exceeding harvesting limits. [214]

Thus, in [253] we aimed at algorithmic methods for extracting geospatial and spatiotemporal activity patterns associated with dark fishing from vessel tracking data recorded over time periods of several years. The results provide input for intelligence based profiling and ranking of known fishing vessels in relation to how frequently individual vessels are linked to observable behavior that diverts from routine operations. In the presence of uncertainty, predictive modeling using knowledge extracted from historical data reduces the size of the search space and increases the chances of disrupting IUU fishing. Even an increase of a few percentage points in the success rate is a lot in absolute numbers when dealing with an enormous fleet of ships and boats that operate across vast coastal areas and open oceans, considering that limited surveillance resources constrain maritime domain awareness and compromise seamless security coverage at all times [213].

Problem Definition. Given a set of fishing vessels $\{v_i\}$ and their history of trajectories $\{\tau_{v_i}\}$, we need to rank the vessels based on their suspiciousness value. This problem definition is novel, especially its application to the domain of IUU fishing. In order to address the problem, we segment the trajectories into end-to-end trips and monitor vessels behavior to detect abnormal activity patterns.

7.2.1 Proposed framework

A summary of the proposed method to solve this problem is as follows. The input is historical tracking data from vessels routinely operating in coastal waters. The output is a ranking that places more specious vessels—those with more frequent abnormal activities—at a higher rank. Naturally, two or more vessels may be tied for the same anomaly score. The ranking used here is based on a weak order. A weak ordering is a mathematical formalization of the intuitive notion of a ranking of a set where some members may be tied with each other [237]. Vessels need to report their operational status in real time, including fishing activity (by broadcasting Status = 7). A non-compliant vessel may misreport the status or may even go dark by turning off its AIS transmitter. Any longer gaps in the transmission of AIS reports indicate that a vessel may intentionally hide its activity.

All that said, we take a conservative approach in using AIS data to detect illegal fishing activities. The ranking differentiates vessels with a distinctive history of frequent abnormal (suspicious) activities from the majority of vessels normally complying with commercial fishing regulations. This is done in three main steps:

Endpoint identification. Fishing vessels do not necessarily stop at major ports, especially when engaging in illicit activities. Thus, we develop our own method for identifying endpoints. This way, we do not miss any information about vessel movements between any two locations. The endpoint identification phase performs two consecutive clustering steps: 1) extraction of anchorage points, and

2) identification of endpoints among the extracted anchorage points. Anchorage points are locations where a vessel remains stationary (within close proximity) for longer periods of time. In the second step, we cluster anchorage points extracted from the trajectories of all fishing vessels to determine proper endpoints. These are common anchorage locations shared by a number of vessels. The list of endpoints includes well-known ports but also nonessential and widely unknown (hidden) harbors.

End-to-end trip segmentation. For a given trip between two identified endpoints, the related AIS reports render a trajectory by associating locations visited with time stamps. Because harvesters generally arrange their fishing activities as a trip, we expect to observe strong patterns across similar trips. AIS data does not directly provide information on vessel trips and analyzing trajectories as a whole may miss patterns in shorter trips, especially when analyzing regions of interest [40]. Noise, corruption, and inconsistencies, as well as 'dark interval' in AIS data translate into inaccurate or missing reports. So an interpolation technique with respect to the curvature of the earth is applied on each trajectory so as to complete missing reports for reasonably short gaps. Then, each interpolated trajectory is segmented into several end-to-end trips. Each such trip begins and ends at an anchorage point which is common among a number of vessels.

Ranking suspicious activity. The obtained trips represent the behavior of vessels with a proper granularity at which one can differentiate normal activities from suspicious ones. The following subsections describe each of the three steps in more detail. We compute two separate anomaly scores for each end-to-end trip: a global anomaly score and a local anomaly score.

Global anomaly score is a measure for the deviation of a vessels behavior for a given trip relative to other vessels behavior for similar trips. Lets assume $T^{v_1}_{a,b}, T^{v_2}_{a,b}, ..., T^{v_n}_{a,b}$ are complete end-to-end trips between endpoints a and b. The length of a trip is the most important indicator for the type of a trip. We apply *Kernel Density Estimation (KDE)* clustering on the length of trips to divide them into a number of clusters, $C_{a,b} = \{c^1_{a,b}, c^2_{a,b}, ..., c^m_{a,b}\}$, such that trips in the same cluster are more similar than those in other clusters. We emphasize that the input of the KDE method is only complete trips as we can learn most common trip types from full trajectories. The global anomaly score of trip $T^{v_1}_{a,b}$ is computed as

$$\mathcal{GAN}(T^{v_1}_{a,b}) = \underset{c^i_{a,b}}{argmin}\,\text{Distance}(L^{v_1}_{a,b}, c^i_{a,b})$$

where $\text{Distance}(L^{v_1}_{a,b}, c^i_{a,b})$ is the distance of data point $L^{v_1}_{a,b}$ from the center of cluster $c^i_{a,b}$.

Local anomaly score measures the compliance of a vessel regarding the trip in question. Lets assume in an end-to-end trip of vessel v_1 between end points a and b, with length of $L^{v_1}_{a,b}$, there are m dark periods longer than the threshold δ_{dark}. Each of these dark periods has the length of $l^{v_1}_i$ and $\mathcal{DP}^v_{a,b} = \{l^{v_1}_1, l^{v_1}_2, ..., l^{v_1}_m\}$. For such an end-to-end trip the local anomaly score is

$$\mathcal{LAN}(T^{v_1}_{a,b}) = \lambda \times \frac{\sum_{i=1}^{m}(l^{v_1}_i)}{L^{v_1}_{a,b}}$$

If a vessel reports the navigational status of engaged in fishing more often than defined threshold times, we label the trip as a *fishing trip*. If a vessel goes dark in an end-to-end trip with fishing activity, it can be interpreted as a more suspicious trip than a non-fishing trip. We reflect on this factor by computing the local anomaly score using parameter λ. We use $\lambda = 1$ for a non-fishing and $\lambda > 1$ for a fishing trip. Trip suspiciousness is simply the summation of global and local anomaly scores

$$\theta_{T^{v_1}_{a,b}} = \alpha \times \mathcal{LAN}(T^{v_1}_{a,b}) + \beta \times \mathcal{GAN}(T^{v_1}_{a,b})$$

where α and β are parameters for adjusting the perceived significance of global and local anomaly scores. Finally, for a given vessel v_1, with observed end-to-end trips $E = \{T^{v_1}_{a_1,b_1}, T^{v_1}_{a_2,b_2}, ..., T^{v_1}_{a_k,b_k}\}$, vessel suspiciousness of v_1 is computed as follows

$$\sigma_{v_1} = \sum_{\forall T_{a_i,b_i} \in E} \theta_{T^{v_1}_{a,b}} \times e^{\frac{-(t-t_{a_i,b_i})}{\rho}}$$

where t is the current time and t_{a_i,b_i} is the start time of the end-to-end trip T_{a_i,b_i} . We amplify the impact of more recent trips compared to older trips using parameter ρ for adjusting how the trip time affects the suspiciousness score.

7.2.2 Experimental results

In this section, we briefly describe spatial patterns of suspicious vessels detected by our method and refer the reader to [253] for thorough details.

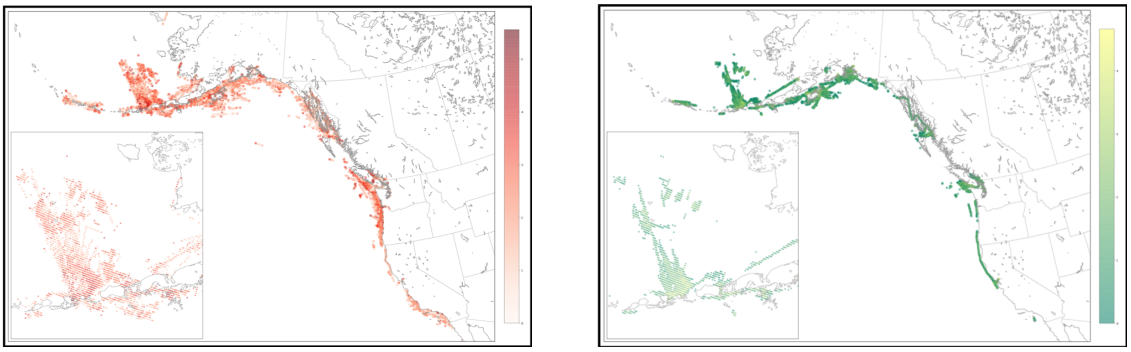
We study the spatial pattern of more suspicious vessels and compare it to the behavior of normal vessels. To do so, we apply a grid to the area of interest, UTM zones 1 to 11. Each grid cell is roughly 10 km on each side.

Figure 7.1-a shows the heatmap of fishing effort of all vessels based on the status of received AIS reports. Each cell indicates the accumulative fishing effort of all vessels happened in that cell.

Figure 7.1-b depicts the heatmap for the density of going dark phenomena of fishing vessels. Each cell shows the summation of being dark by all vessels. It is worth mentioning, the color density of each cell corresponds to the length of the period where the vessels start going dark from.

Considering the overall behavior of vessels, we divide the cells into three sets: white cells, gray cells, and black cells. White cells are the ones in which the majority of the observed vessels have normal behavior. A cell is considered white if more than 70% of the observed vessels always send their AIS messages. A cell is gray if the percentage of the vessels sending their AIS message is between 70% and 30%. And a cell is black if less than 30% of vessels always transmit their AIS messages. In other terms, in a black cell the majority of vessels have been dark at least once.

A white cell can be in an area that vessels do not have any excuse not to transmit the AIS messages since a large set of vessels were able to do so. In a gray cell, we see a mix of behaviors from vessels where a meaningful percentage of vessels send their AIS messages and a similar percentage of vessels did not. A good example of black cells can be areas with weak signal. In each cell, we closely



(a) (b)

Figure 7.1: a) Heatmap of Going Dark Area Based on Received AIS Reports, b) Heatmap of Fishing Density Based on Revised AIS Reports

study vessels activities considering their suspiciousness. To do so, we divide vessels into two groups based on their suspiciousness score: vessels in top 50% of the list as more suspicious vessels and the remaining as less suspicious vessels. This can be considered as a binary classification of vessels in terms of their suspiciousness score.

In 82% of white cells, we see that more suspicious vessels have similar behavior by turning off their AIS transponder at least one time. In other words, in 82% of white cells, each vessel that went dark was in the list of more suspicious vessels. This result is 77% and 68% for the gray and black cells, respectively. This means for the remaining vessels, contribution of global anomaly score is greater to value of their vessel suspiciousness score. And now we observe, on average in 79% of all cells, all of the more suspicious vessels go dark. This is a strong behavioral pattern which shows that the proposed ranking algorithm was successful in the distinction of less and more suspicious vessels.

7.3 Fishing Vessels Activity Detection from Longitudinal AIS Data

Motivation. Maritime domain awareness calls for continuous monitoring and tracking of fisheries using data and information from maritime intelligence sources to detect illegal fishing activities. Vessel tracking services routinely gather data for identifying, locating, and capturing information about ships and boats operating in coastal waters and across open oceans. Marine traffic data in the form of multi-variate time series is interpreted as *trajectories* of vessel movements over time periods ranging from a few hours to several years. The total data volume renders manual processing impossible, raising an immediate need for autonomous and smart systems to follow the vessels' footprints in near real-time and detect their basic activity types. As discussed in Section 7.2, Fisheries authorities and regional fisheries management organizations face many difficulties in detecting and combating IUU fishing. One of the fundamental obstacles is the lack of systematic methods for measuring vessels fishing effort. In the presence of uncertainty, predictive modeling, trained using

expert-labelled data, reduces the size of the search space and increases the chances of disrupting IUU fishing.

Problem Definition. We define the fishing activity detection problem and the scope of the proposed solution in abstract terms as a binary classification model for interpreting trajectories extracted from marine traffic data. Let $v \in V$ denote a *vessel* with AIS equipment which can engage in fishing activities. Further, let τ_v denote a temporally ordered set of AIS reports, $\{r_{\tau_v}\}$, from a trajectory associated with vessel v. A report $_{\tau_v}$ is a vector of values for the AIS features. We then formulate our problem as follows:

> Given a set of fishing vessels $V = \{v_1, v_2, \ldots, v_m\}$ and their corresponding trajectories $T = \{\tau_{v_1}, \tau_{v_2}, \ldots, \tau_{v_m}\}$, fishing activity detection is the task of determining if a vessel $v_i \in V$ at time t was engaged in fishing or not by analysing δ consecutive reports from τ_{v_i}. This problem can be extended to identification of the whole trajectory representing an end-to-end fishing activity.

The notation $\tau_{v_i,t}$ denotes a specific time point t in a trajectory τ of some vessel v_i.

7.3.1 Proposed framework

A summary of the proposed method, called *FishNET* [14], to solve the mentioned problem is described in this section.

FishNET overcomes the trajectory classification challenges by utilizing a 4-stage algorithm to differentiate fishing activities from non-fishing ones. The detection framework is illustrated in Figure 7.2. First, the *Trajectory Reconstruction* component identifies the vessels' footsteps and generates a set of motion-related features to produce trajectories invariant to location and time. The next component, *Segment Generation*, prepares the classifier's input by utilizing a sliding window segmentation to partition trajectories into short segments, which are labeled based on their likelihood of showing fishing activity. In the end, the vessel status at each temporal point will be determined based on the majority labels assigned to the segments including the given point; this facilitates the end-to-end fishing trajectory identification process.

Trajectory reconstruction

The trajectory reconstruction process handles a variety of challenges we are facing in preparing the data for the predictive analysis. For example, the data is exposed to different noise sources, like infeasible speed or location, which are cleaned or replaced. Also, the time sampling for AIS reports is sometimes irregular. To reconstruct regular trajectories of the message sequences, we apply linear spatial interpolation relying on the *constant velocity* assumption for fishing attempts. However, the granularity level of reconstructed trajectories matters. Based on our analysis, resampling trajectories at a 10-minute time scale is logical, because it does not leave a big gap between two successive AIS reports, which helps to avoid potential interpolation errors.

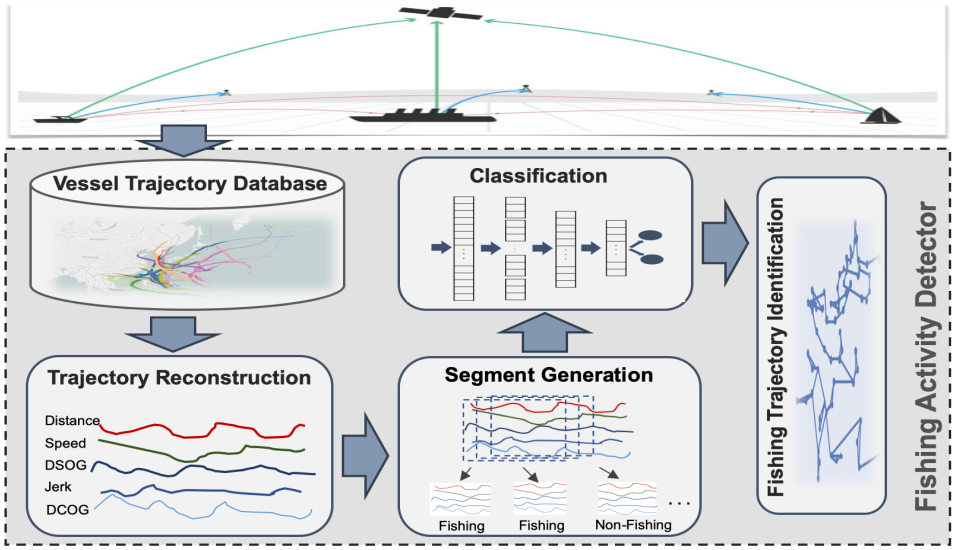


Figure 7.2: *FishNET*: Fishing activity detection framework

A generalizable trajectory classifier is expected to find the movement patterns and trajectory shapelets independent of the exact time and space. In other words, we expect that shipping trajectories share the same basic sub-patterns inrrespective of geospatial characteristics. However, classifying trajectories based on the original features included in AIS reports, might lead the model to develop some undesired decision logic based on a vessel's location or fishing activity time. We thus transform the original features in the trajectories of each vessel, $\tau_{v,t}$, into a set of motion-related features $\zeta_{v,t}$ that are suitable for discovering dynamic behavioral patterns of fishing activities independent of spatiotemporal details.

$$\{\tau_{v,t}\}_{t=0}^{n} \Rightarrow \{\zeta_{v,t}\}_{t=0}^{n} \tag{7.1}$$

Generally, vessel motions as shown in Figure 7.3 are defined by six degrees of freedom: three components of translation (heave, sway, surge) and three components of rotation (pitch, roll, yaw) [210]. On this basis, the following set of performance and movement related characteristics can be extracted [200], as thoroughly described in [14]: *Distance* ($\mathbf{D}_{v,t}$), *Rectilinear speed* ($\mathbf{S}_{v,t}$), *Rectilinear acceleration* ($\mathbf{A}_{v,t}$), *Rectilinear Jerk* ($\mathbf{J}_{v,t}$), *Derivative of course* ($\Psi_{v,t}$).

Thus, $\zeta_{v,t}$ is defined based on $\{\mathbf{D}_{v,t}, \mathbf{S}_{v,t}, \mathbf{A}_{v,t}, \mathbf{J}_{v,t}, \Psi_{v,t}\}$ as the moving behavioral characteristics of fishing vessels. Applying these intrinsic attributes as predictive features not only makes *FishNET* robust to differences in geographical locations and timestamps of trajectories but also eliminates the need for considering the fishing vessel types in the classification process.

Segment generation

As discussed earlier, fishing activity routinely extends over considerable time periods from several hours to days based on the fishing type [68]. Feeding a classifier model with such very long input sequences may not only misguide it and decrease the performance but will also increase the model's

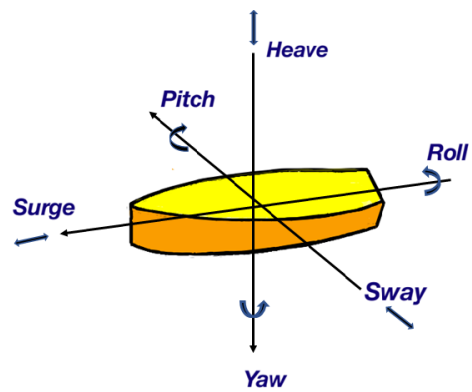


Figure 7.3: Vessel motion: six degrees of freedom

complexity and training time. Thus, to prepare the classifier's input, we extract trajectory fragments $\mathbf{w}_t$ out of the transformed trajectory ζ_t using a sliding window with size δ that moves across the signals and generates a set of fixed-length sequences.

$$\{\zeta_{v,t}\}_{t=1}^{n} \Rightarrow \{\{\mathbf{w}_t\}_{t=\delta}^{n} \mid \mathbf{w}_{v,t} = \{\zeta_{v,i}\}_{i=t-\delta-1}^{t}\} \tag{7.2}$$

Based on our experiments, on multiple values of δ we found that the setting of $\delta = 11$ helps the classifier to achieve the highest accuracy level, which intuitively makes sense; a sequence of about 2 hours is long enough to be informative and representative of a vessel's movement behavior.

Next, a majority voting scheme is applied to assign a class label to the generated sequences. However, as the fishing vessels do not frequently switch their status, the majority of window sequences are pure fishing or non-fishing cases. To simplify labelling the entire sequence based on the majority rule,we considered odd values for δ.

CNN-based classification

To classify the fixed-length multivariate segments generated in the previous phase, we design a one-dimensional convolutional neural network (1D-CNN). The output of this model is a binary label β assigned to any arbitrary input segment $\mathbf{w}$ to determine its *fishing/non-fishing* status:

$$\text{Classifier}(\mathbf{w}) \rightarrow \beta \tag{7.3}$$

The main reason for using a 1D-CNN is its significant performance in recent studies and its fairly low computational complexity. The model consists of two main modules: convolution and classification. The convolution module itself includes two main types of layers: *convolutional* and *pooling* layers. The former extracts in-depth features from the input sequences, while the latter filters the extracted features to highlight the most salient elements. In the end, the classification module, containing a set of fully connected layers, learns from the constructed internal representation how to classify the input segments.

Thus, as the kernel, $\text{conv}\,1D(.,.)$, in our $\text{Classifier}_{\text{CNN}}$ slides along with the 2D inputs w, it extracts the appropriate artificial features to distinguish fishing trajectories from the rest. Thus, as the Equation 7.4 shows, the input is convolved with the kernels and is executed through an activation function to produce the first layer's output feature map.

$$\mathbf{o}_j^l = b_j^l + \sum_{i=1}^{N_{l-1}} \text{conv}\,1D\left(\mathbf{k}_{ij}^{l-1}, \mathbf{w}_i^l\right) \mid l = 1 \tag{7.4}$$

In general, the same process is followed through the next layers [34, 156]:

$$\mathbf{o}_j^l = f\left(\sum_{i \in M_j} \mathbf{o}_i^{l-1} * \mathbf{k}_{ij}^{l-1} + b_j^l\right) \mid l \in 2..L \tag{7.5}$$

where $\mathbf{w}_i^l$ is defined as the ith element of input, L determines the number of convolutional layers, b_j^l is defined as the bias of the jth neuron at layer l, $\mathbf{o}_j^l$ is the output of the ith neuron at layer $l-1$, $\mathbf{k}_{ij}^{l-1}$ is the kernel from the ith neuron at layer $l-1$ to the jth neuron at layer l.

In the next step, the final output of the convolution module is fed into the fully connected layers to map them to the final outputs of the network by determining the probability distribution over the set of target classes for the given input.

$$\mathcal{FCN}(\mathbf{o}^l) \rightarrow \alpha \mid l = L \tag{7.6}$$

Finally, the class with the maximum probability will be selected as the segment's *fishing/non-fishing* status ($\max(\alpha) \rightarrow \beta$).

Fishing trajectory identification

Identifying the whole trajectory representing an end-to-end fishing activity is the ultimate goal of *FishNET*. First, point-wise labeling is performed based on the majority voting approach (Equation 7.7), which determines the fishing likelihood in the given point. The voting algorithm decides based on the label assigned by the classifier to all the δ segments which include the given point.

$$\text{C}_{\text{MV}}(\zeta_{v,t}) = \text{argmax}_i\, \Sigma_{j=\delta-1}^{0} \mathbf{I}(\text{Classifier}(\zeta_{v,\{(t-j)..(t+\delta-j)\}})) \tag{7.7}$$

$\zeta_{v,t}$ is a specific point of a vessel trajectory, $\zeta_{v,t_1..t_2}$ are the generated segments which include $\zeta_{v,t}$, and $I(.)$ is an indicator function. Then, each end-to-end fishing activity belonging to vessel v will be spotted based on uninterrupted fishing intervals.

$$\begin{aligned}&\{\zeta_{v,i}, ..., \zeta_{v,j} \in \mathbf{FT} \mid \forall seq = \zeta_{v,k}..\zeta_{v,m} \mid k \geq i, m \leq j, size(seq) \geq 2 * \theta \\ &\Rightarrow \text{argmax}(\text{C}_{\text{MV}}(\zeta_{v,t}) \in seq) == \textit{fishing}\}\end{aligned} \tag{7.8}$$

θ indicates the length of permitted *non-fishing* gaps in the identified trajectory, and **FT** determines the set of end-to-end trajectories involved in fishing activities.

7.3.2 Experimental results

For this project, we use a labelled dataset [3] to train our model and two unlabelled datasets to run longitudinal analysis:

- *U. S. AIS Dataset.* The AIS data provided by the U. S. Coast GuardServices (available at www.marinecadastre.gov) covers all of the coastal waters of the United States and most of Canada from 2015 through 2018.
- *Denmark AIS Dataset.* The AIS data provided by the Danish shore-based AIS system (available at dma.dk) covers vessel movement in Denmark's coastal waters for the period of 2015-2018.

The performance of *FishNET* is scrutinized in [14], and its results are compared with other *GFW*, to the best of our knowledge the only among the state-of-the-art of fishing activity detection methods evaluated using large real-world data. As the analysis show *FishNET* is able to outperform GFW in some detecting fishing pattern of some vessel types. The following two subsections present the temporal and anomaly detection related analysis of *FishNET*.

Temporal distribution of fishing effort

Complexity of fisheries management due to the heterogeneity of this phenomenon. In addition to spatial, temporal variability is also playing an important role in catch size and fishing effort distribution. Analyzing vessels activity at a high temporal resolution makes our image of global fishing more complete. Figures 7.4a and 7.4b show the temporal distribution of fishing effort for the sea around Denmark and North Americas Coastal Areas, respectively, for the studied period. Comparing the fishing effort pattern in both regions, we can see a similar trend: lower level of fishing in colder months of the year and a higher level of fishing effort in warmer months of the year, as expected. In comparing of yearly temporal fishing effort distributions of the sea around Denmark, we can see many similar but not identical patterns. Interestingly, the yearly fishing effort distributions of North America's coastal area have a strong similarities.

Anomalous fishing activity detection

As a practical application of *FishNET* here, we discuss how successful fishing activity detection can be used to overcome this difficulty. The output of *FishNET* provides input for intelligence-based profiling and ranking of vessels linked to observable behavior that diverts from routine operations.

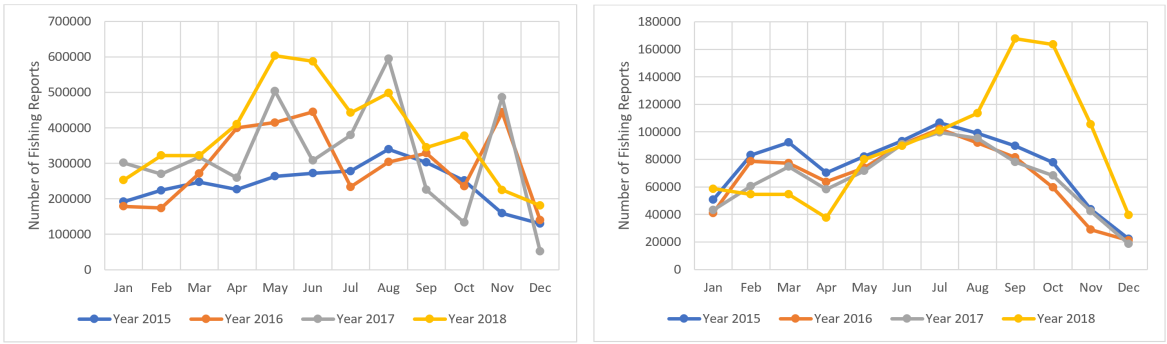
A small improvement in the success rate of quantification and identification of IUU fishing means a lot in absolute numbers when dealing with an enormous fleet of ships and boats that operate across vast coastal areas and open oceans, considering the limited surveillance resources constrain maritime domain awareness and security [112]. The very first group of vessels susceptible to IUU

fishing are the ones in the top of a list ordered by fishing hours. Here, we focus on the top 10% of most active fishing vessels. In the sea around the Denmark area, the top 10% of most active vessels are fishing 1791 hours on average six times more than an average vessel in this area. 153, 160, 177, and 142 vessels are in this list for the years 2015, 2016, 2017, and 2018 respectively. There are only 10 vessels which are continuously active for all the four years. Among these 10 vessels, 7 of them are industrial fishing vessels with the length of more than 20 meters, and more specifically 4 of them are trawlers. While continuous fishing activity is assumptive, such phenomenon for a small vessel is strange and needs close investigation by monitoring agencies. Although the U.S. dataset contains a fewer number of vessels, the average hours of fishing for the top 10% most active fishing vessels is 1490 hours. 38, 35, 35, and 44 vessels are in this list for the years 2015, 2016, 2017, and 2018, respectively. Only three vessels are on this list for all four years.

Fishing in marine protected areas can cause significant threats to already endangered species. To extract patterns of fishing in rare spots, we compute for each grid the ratio of the total fishing hours to the number of vessels for that grid. Grids with a higher ratio are the ones with more fishing done by a fewer number of vessels and need closer investigation. In the sea around Denmark, we found 45 areas with a high ratio, seven of them are relatively far from the shoreline, which makes them even more suspicious. In North American coastal waters, we found 61 areas, four of these are further from the shoreline, and 3 of them are around Alaska. As mentioned earlier such analysis can be more meaningful and applicable in collaboration with experts monitoring fishing activity.

7.4 Conclusions

Spatiotemporal data differs from the analyzed time series in the previous sections and brings its own challenges to deal with. Our spatiotemporal related projects target marine safety and maritime domain awareness challenges. Illegal, unreported, and unregulated fishing gravely threatens the sustainability of fisheries, marine ecosystems, and fish populations worldwide. Constrained by limited surveillance resources, enforcement agencies face immense challenges in their efforts to gain the upper hand in



(a) The seas around Denmark

(b) North Americas coastal areas

Figure 7.4: Temporal distribution of fishing effort

dealing with an enormous fleet of fishing boats and ships operating across vast coastal areas and open oceans, in total covering more than 70 percent of our planets surface.

The proposed techniques in this section for tracking dark fishing by profiling and ranking fishing vessels aims at enhancing efficiencies through predictive models that utilize historical data and information which is available in abundance. Narrowing down the search space by drawing on probability distributions for suspected dark fishing leads to more targeted procedures for checking and probing vessels and their catches.

Chapter 8

Lessons Learned and Future Work

This chapter lays out the final results and the future scope for the problems addressed in this dissertation.

8.1 Lessons Learned

The very nature of large-scale critical infrastructure necessitates advanced analytical methods that contribute to situational awareness of the decision-maker, specifically to ensure that they know what is happening in the problem domain. While it is impossible (or even unnecessary) to be aware of everything, especially in large-scale systems, it is nonetheless crucial to be aware of the events of interest. Therefore, the present study focuses on proposing novel algorithms in an end-to-end framework of anomaly/event detection in stream data. Accordingly, we exemplified and experimentally validated the framework proposed herein for critical cyber-physical systems. Nevertheless, the proposed approaches are conceptually applicable to a variety of other real-world domains.

Despite all existing techniques, several fundamental challenges related to situation analysis in stochastic stream data must be addressed. Thus, as illustrated in Figure 1.1, the present work explicates the steps required to achieve a scalable and interpretable anomaly detection framework, which are summarized below.

First, we proposed an efficient and interpretable anomaly-detection algorithm to trace suspicious activities in SCADA-based water supply systems. To this end, the existing anomaly-scoring problem of HMM techniques is solved. In addition, the trade-off between selecting the right input window size and granularity is addressed by presenting a hierarchical multi-granular model.

Furthermore, to overcome the limitations of behavioral modeling, a time-series modeling method with a high average accuracy but a small error deviation is required. Accordingly, we proposed a new hybrid method capable of handling complicated time series consisting of both linear and nonlinear components. The proposed framework also handles the problem of noisy data by using novel multi-branch data augmentation.

Then, to address challenges related to anomaly scoring, we devised an attack-detection model that functions through a distributed detector network. In particular, applying dynamic attack scoring boosts the analytic performance of the proposed method to efficiently detect collective attacks by orchestrating a network of detectors and reduces false-alarm rates by ignoring potential contextual noise and errors in the applied behavioral predictors.

The ultimate outcome of the entire framework must satisfy the objectives and expectations of real-world domain experts. Given the immense importance of having an effective anomaly detection method, the interpretability of its decision-making process is likewise important. Therefore, a model-agnostic explainer is proposed to fill the black-box gap and provide the required interpretability to both convince the human analyst about the isolating factors and highlight the models potential limitations. Furthermore, some opportunities of model-specific explainers and causal profiling-based XAI are explored. This step not only builds trust by involving the human analyst in the predictive process, but it can also grow AI for common good and help both critical and noncritical organizations ensure that the decisions made by their applied AI are ethical.

In addition, spatiotemporal data analysis was explored with a focus on maritime domain awareness. Accordingly, in a joint research, we proposed two novel methods for trajectory classification and anomaly detection to address the catastrophic issue of dark or illegal fishing.

It should be noted that this study was based on several years of collaborative, funded safety and security research projects with industry and academic partners. Accordingly, we strongly believe the concepts and findings reported herein can inspire new research and development directions with applications in cyber intelligence, stream data analysis, and anomaly detection as well as decision-making supports for other critical security operations. Nevertheless, several problems and challenges in the realm of end-to-end anomaly detection in stream data remain to be addressed.

8.2 Future Work

I briefly outline a number of directions for future research in the field of temporal anomaly detection herein.

- **Improving anomaly scoring techniques.** The efficacy of the methods introduced and proposed in the anomaly-scoring phase of this end-to-end framework can be improved and extended in numerous ways. It would be of interest to develop more sophisticated methods based on reinforcement learning that can verify the systems impact and enhance the scoring phase. Moreover, other complex algorithms and mechanisms for computing the confidence, significance, and impact of a suspicious event can be developed.

- **Improving isolation explainability.** There are a variety of future directions to improve isolation explainability. For example, it would be of interest to provide qualitative specifications and formal explanations of the quantitative specifications learned by the statistical models that utilize logical AI. The main advantages of formal logic are internal property transparency and

the possibility of interpretable representation in different levels of abstraction. Other possibilities in this area include developing more-sophisticated but transparent situation-visualization and alert-generation mechanisms.

- **Enhancing test platform.** Given the importance of validation and testing in model development, test platforms can be improved. A lack of labeled data and difficulty generating a realistic time series with anomalies make time-series anomaly detection very challenging. Thus, generative adversarial networks can be used to produce realistic but domain-dependent data with various challenging anomalies.

- **Transferable time series anomaly detection.** The present findings can be extended to transferable time-series anomaly detection. It is often difficult to collect a large set of high-quality or labeled datasets to train advanced anomaly detection models from scratch. Thus, transfer learning and domain adaptation can reduce training time, enabling a high-performance model to be applied to a different domain or task. Accordingly, transfer learning in the context of time-series anomaly detection is another promising area that has received little attention. Regardless, devising an effective transfer method will facilitate the analytical process and enable anomaly detection to be treated as a supervised learning problem.

Bibliography

[1] Global threat landscape report: A semiannual report by fortiguard labs. https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/threat-report-h1-2020.pdf. Accessed: 2020-10-25.

[2] Outlier detection for temporal data. https://archive.siam.org/meetings/sdm13/gupta.pdf. Accessed: 2018-09-30.

[3] Sustainability through transparency. global fishing watch. https://globalfishingwatch.org/.

[4] Time series anomaly detection algorithms. https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2. Accessed: 2018-10-20.

[5] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE, 2012.

[6] Aderemi O Adewumi and Andronicus A Akinyelu. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, 8(2):937–953, 2017.

[7] United States Environmental Protection Agency. Cyber security 101 for water utilities. Accessed: July 2016.

[8] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[9] Musse Mohamud Ahmed and WL Soo. Supervisory control and data acquisition system (scada) based customized remote terminal unit (rtu) for distribution automation system. In *2008 IEEE 2nd International Power and Energy Conference*, pages 1655–1660. IEEE, 2008.

[10] Aijun An, Christine Chan, Ning Shan, Nick Cercone, and Wojciech Ziarko. Applying knowledge discovery to predict water-supply consumption. *IEEE Expert*, 12(4):72–78, 1997.

[11] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015.

[12] Jerone Andrews, Thomas Tanay, Edward J Morton, and Lewis D Griffin. Transfer representation-learning for anomaly detection. JMLR, 2016.

[13] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Discovering characterizations of the behavior of anomalous subpopulations. *IEEE Transactions on knowledge and data engineering*, 25(6):1280–1292, 2012.

[14] Saeed Arasteh, Mohammad A. Tayebi, Zahra Zohrevand, Uwe Glässer, Amir Yaghoubi S., Parvaneh Saeedi, and Hans Wehn. Fishing vessels activity detection from longitudinal ais data. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20)*, 2020.

[15] Barak Ariel and David P Farrington. Randomized block designs. In *Handbook of quantitative criminology*, pages 437–454. Springer, 2010.

[16] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[17] Mikhail Atallah, Wojciech Szpankowski, and Robert Gwadera. Detection of significant sets of episodes in event sequences. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 3–10. IEEE, 2004.

[18] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-temporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)*, 51(4):1–41, 2018.

[19] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Technical report, 2000.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[21] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[22] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, 2007.

[23] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD explorations newsletter*, 2(2):81–85, 2000.

[24] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

[25] Irad Ben-Gal. Outlier detection. In *Data mining and knowledge discovery handbook*, pages 131–146. Springer, 2005.

[26] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[27] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[28] Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, and Cherita L Corbett. A survey of deep learning methods for cyber security. *Information*, 10(4):122, 2019.

[29] Renaud Bidou. Security operation center concepts & implementation. *avalable at http://www. iv2-technologies. com*, 2005.

[30] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*, 2020.

[31] Hans-Peter Blossfeld and Götz Rohwer. Causal inference, time and observation plans in the social sciences. *Quality and quantity*, 31(4):361–384, 1997.

[32] Alisa Bokulich. Searching for noncausal explanations in a sea of causes. *Explanation Beyond Causation: Philosophical Perspectives on Non-Causal Explanations*, page 141, 2018.

[33] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[34] Jake Bouvrie. Notes on convolutional neural networks. *In Practice*, pages 47–60, 2006.

[35] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[36] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[37] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[38] Robert A Bridges, Jessie D Jamieson, and Joel W Reed. Setting the threshold for high throughput detectors: A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors. *arXiv preprint arXiv:1710.09422*, 2017.

[39] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

[40] Yingyi Bu, Tat-Wing Leung, Ada Wai-Chee Fu, Eamonn Keogh, Jian Pei, and Sam Meshkin. Wat: Finding top-k discords in time series database. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 449–454. SIAM, 2007.

[41] Suratna Budalakoti, Ashok N Srivastava, Ram Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. 2006.

[42] Suratna Budalakoti, Ashok N Srivastava, and Matthew Eric Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):101–113, 2008.

[43] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[44] J Peter Burman and Mark C Otto. Census bureau research project: Outliers in time series. *Bureau of the Census, SRD Res. Rep. CENSUS/SRD/RR-88114*, 1988.

[45] João BD Cabrera, Lundy Lewis, and Raman K Mehra. Detection and classification of intrusions and faults using sequences of system calls. *Acm sigmod record*, 30(4):25–34, 2001.

[46] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016.

[47] Francesco Cauteruccio, Giancarlo Fortino, Antonio Guerrieri, Antonio Liotta, Decebal Constantin Mocanu, Cristian Perra, Giorgio Terracina, and Maria Torres Vega. Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance. *Information Fusion*, 52:13–30, 2019.

[48] Sezen Cekic, Didier Grandjean, and Olivier Renaud. Time, frequency, and time-varying granger-causality measures in neuroscience. *Statistics in medicine*, 37(11):1910–1931, 2018.

[49] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural networks*, 5(6):961–970, 1992.

[50] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[51] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[52] Varun Chandola, Varun Mithal, and Vipin Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE international conference on data mining*, pages 743–748. IEEE, 2008.

[53] Rohitash Chandra and Mengjie Zhang. Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction. *Neurocomputing*, 86:116–123, 2012.

[54] Ioannis Chapsos and Steve Hamilton. Illegal fishing and fisheries crime as a transnational organized crime in indonesia. *Trends in Organized Crime*, 22(3):255–273, 2019.

[55] Snehamoy Chatterjee, Ansuman Dash, and Sukumar Bandopadhyay. Ensemble support vector machine algorithm for reliability estimation of a mining machine. *Quality and Reliability Engineering International*, 31(8):1503–1516, 2015.

[56] Lu Chen, Hongya Qiu, Junhong Zhang, Vijay P Singh, Jianzhong Zhou, and Kangdi Huang. Copula-based method for stochastic daily streamflow simulation considering lag-2 autocorrelation. *Journal of Hydrology*, 578:123938, 2019.

[57] Tao Cheng and Zhilin Li. A multiscale approach for spatio-temporal outlier detection. *Transactions in GIS*, 10(2):253–263, 2006.

[58] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[59] Jared Connell and Søren Højsgaard. Hidden semi markov models for multiple observation sequences: The mhsmm package for r. *Journal of Statistical Software*, 39(4):1–22, 2011.

[60] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[61] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[62] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[63] Pádraig Cunningham, Dónal Doyle, and John Loughrey. An evaluation of the usefulness of case-based explanation. In *International Conference on Case-Based Reasoning*, pages 122–130. Springer, 2003.

[64] Xuan Hong Dang, Barbora Micenková, Ira Assent, and Raymond T Ng. Local outlier detection with interpretation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 304–320. Springer, 2013.

[65] Dipankar Dasgupta and Nivedita Sumi Majumdar. Anomaly detection in multidimensional data using negative selection algorithm. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1039–1044. IEEE, 2002.

[66] Dipankar Dasgupta and Fernando Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 1, pages 125–130. IEEE, 2000.

[67] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[68] Erico N de Souza, Kristina Boerder, Stan Matwin, and Boris Worm. Improving fishing pattern detection from satellite ais using data mining and machine learning. *PloS one*, 11(7):e0158248, 2016.

[69] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[70] Hannah M Dee and David C Hogg. Detecting inexplicable behaviour. In *BMVC*, pages 1–10, 2004.

[71] Konstantinos Demertzis and Lazaros Iliadis. A computational intelligence system identifying cyber-attacks on smart energy grids. In *Modern Discrete Mathematics and Analysis*, pages 97–116. Springer, 2018.

[72] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[73] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[74] UC Berkeley EECS Dept. Cyber-physical systems. `cyberphysicalsystems.org`, 2020. accessed: 05-January-2020.

[75] Patrik D'haeseleer, Stephanie Forrest, and Paul Helman. An immunological approach to change detection: Algorithms, analysis and implications. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 110–119. IEEE, 1996.

[76] Thomas G Dietterich. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[77] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[78] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[79] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[80] Erol Egrioglu, Cagdas Hakan Aladag, and Ufuk Yolcu. Fuzzy time series forecasting with a novel hybrid approach combining fuzzy c-means and neural networks. *Expert Systems with Applications*, 40(3):854–857, 2013.

[81] Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158*, 2015.

[82] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pages 16–21. ACM, 2013.

[83] David Endler. Intrusion detection. applying machine learning to solaris audit data. In *Proceedings 14th Annual Computer Security Applications Conference (Cat. No. 98EX217)*, pages 268–279. IEEE, 1998.

[84] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[85] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160, 2009.

[86] Eleazar Eskin, Wenke Lee, and Salvatore J Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 165–175. IEEE, 2001.

[87] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[88] Paul F Evangelista, Piero Bonnisone, Mark J Embrechts, and Boleslaw K Szymanski. Fuzzy roc curves for the 1 class svm: Application to intrusion detection. In *Proceedings International Joint Conference on Neural Networks, IJCNN*. Citeseer, 2005.

[89] Durdu Ömer Faruk. A hybrid neural network and arima model for water quality time series prediction. *Engineering Applications of Artificial Intelligence*, 23(4):586–594, 2010.

[90] Wenhui Feng and Chongzhao Han. A novel approach for trajectory feature representation and anomalous trajectory detection. In *Information Fusion (Fusion), 2015 18th International Conference on*, pages 1093–1099. IEEE, 2015.

[91] Erik M Ferragut, Jason Laska, and Robert A Bridges. A new, principled approach to anomaly detection. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 210–215. IEEE, 2012.

[92] Fisheries and Oceans Canada. The state of world fisheries and aquaculture 2018. food and agriculture organization of the united nations.

[93] Fisheries and Oceans Canada. The state of world fisheries and aquaculture. contributing to food security and nutrition for all. `http://www.fao.org/3/a-i5555e.pdf`. Accessed: August 2019.

[94] Fisheries and Oceans Canada. Fisheries and oceans canada. illegal, unreported and unregulated. `http://www.dfo-mpo.gc.ca/international/isu-iuu-eng.htm`, 2019. Accessed: August 2019.

[95] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. A sense of self for unix processes. In *in Proc. IEEE Symp. Security Privacy*, page 120128, 1996.

[96] Edward W Frees. *Regression modeling with actuarial and financial applications*. Cambridge University Press, 2009.

[97] Ben D Fulcher, Max A Little, and Nick S Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, 10(83):20130048, 2013.

[98] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[99] Bo Gao, Hui-Ye Ma, and Yu-Hang Yang. Hmms (hidden markov models) based on anomaly intrusion detection method. In *Proceedings. International Conference on Machine Learning and Cybernetics*, volume 1, pages 381–385. IEEE, 2002.

[100] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[101] Ashutosh Garg, Sreeram Balakrishnan, and Shivakumar Vaithyanathan. Asynchronous hmm with applications to speech recognition. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–1009. IEEE, 2004.

[102] Yong Ge, Hui Xiong, Zhi-hua Zhou, Hasan Ozdemir, Jannite Yu, and Kuo Chu Lee. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1733–1736. ACM, 2010.

[103] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[104] André Gensler, Janosch Henze, Bernhard Sick, and Nils Raabe. Deep learning for solar power forecasting—an approach using autoencoder and lstm neural networks. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002858–002865, 2016.

[105] Felix A Gers, Jürgen A Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[106] Anup K Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *USENIX security symposium*, volume 99, page 12, 1999.

[107] Anup K Ghosh, Aaron Schwartzbard, and Michael Schatz. Learning program behavior profiles for intrusion detection. In *Workshop on Intrusion Detection and Network Monitoring*, volume 51462, pages 1–13, 1999.

[108] Eric Ghysels, Clive WJ Granger, and Pierre L Siklos. Is seasonal adjustment a linear or nonlinear data-filtering process? *Journal of Business & Economic Statistics*, 14(3):374–386, 1996.

[109] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*, pages 88–99. Springer, 2016.

[110] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[111] Fabio A González and Dipankar Dasgupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, 2003.

[112] Dana A Goward. Maritime domain awareness: The key to maritime security. *Legal challenges in maritime security. Leiden: Martinus Nijhoff*, 2008.

[113] B Graham. The intelligent investor: The classic bestseller on value investing, 1986.

[114] Edward R Griffor, Christopher Greer, David A Wollman, and Martin J Burns. Framework for cyber-physical systems: Volume 1, overview. 2017.

[115] Christopher Grimsley. Causal and non-causal explanations of artificial intelligence. 2020.

[116] Philip Gross, Janak Parekh, and Gail Kaiser. Secure selecticast for collaborative intrusion detection systems. In *Proceedings of the 3rd International Workshop on Distributed Event-Based Systems (DEBS04)*. IET, 2004.

[117] Water Sector Coordinating Council Cyber Security Working Group et al. Water security roadmap to secure control systems in the water sector, 2008.

[118] Aditya Grover, Ashish Kapoor, and Eric Horvitz. A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 379–386. ACM, 2015.

[119] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93, 2019.

[120] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

[121] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 364–379. ACM, 2018.

[122] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[123] Manish Gupta, Jing Gao, Yizhou Sun, and Jiawei Han. Integrating community matching and outlier detection for mining evolutionary community outliers. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 859–867. ACM, 2012.

[124] Nikhil Gupta, Dhivya Eswaran, Neil Shah, Leman Akoglu, and Christos Faloutsos. Beyond outlier detection: Lookout for pictorial explanation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 122–138. Springer, 2018.

[125] Robert Gwadera, Mikhail J Atallah, and Wojciech Szpankowski. Reliable detection of episodes in event sequences. *Knowledge and Information Systems*, 7(4):415–437, 2005.

[126] James Douglas Hamilton. *Time series analysis*. Princeton university press, 2020.

[127] Junwei Han, Dingwen Zhang, Gong Cheng, Nian Liu, and Dong Xu. Advanced deep-learning techniques for salient and category-specific object detection: a survey. *IEEE Signal Processing Magazine*, 35(1):84–100, 2018.

[128] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[129] David J Hill and Barbara S Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010.

[130] Raymond C Borges Hink, Justin M Beaver, Mark A Buckner, Tommy Morris, Uttam Adhikari, and Shengyi Pan. Machine learning for power system disturbance and cyber-attack discrimination. In *7th international symposium on resilient control systems (ISRCS)*, pages 1–8. IEEE, 2014.

[131] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[132] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[133] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[134] En-Yu Hsu, Chien-Liang Liu, and Vincent S Tseng. Multivariate time series early classification with interpretability using deep learning and attention mechanism. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 541–553. Springer, 2019.

[135] Jianming Hu, Jianzhou Wang, and Guowei Zeng. A hybrid forecasting approach applied to wind speed time series. *Renewable Energy*, 60:185–194, 2013.

[136] Svend Hylleberg. *Modelling seasonality*. Oxford University Press, 1992.

[137] Aapo Hyvärinen, Patrik O Hoyer, and Mika Inki. Topographic independent component analysis. *Neural computation*, 13(7):1527–1558, 2001.

[138] Boris Iglewicz and David Caster Hoaglin. *How to detect and handle outliers*, volume 16. Asq Press, 1993.

[139] Jun Inoue, Yoriyuki Yamagata, Yuqi Chen, Christopher M Poskitt, and Jun Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 1058–1065. IEEE, 2017.

[140] Hesam Izakian and Witold Pedrycz. Anomaly detection and characterization in spatial time series data: A cluster-centric approach. *IEEE Transactions on Fuzzy Systems*, 22(6):1612–1624, 2014.

[141] Vidya Jadhav and Prakash Devale. Anomaly detection on user browsing behaviors for prevention app_ddos. *International Journal of Advances in Engineering & Technology*, 1(5):492, 2011.

[142] HV Jagadish, Nick Koudas, and S Muthukrishnan. Mining deviants in a time series database. In *VLDB*, volume 99, pages 7–10, 1999.

[143] Vikramaditya Jakkula and Diane J Cook. Anomaly detection using temporal data mining in a smart home environment. *Methods of information in medicine*, 47(1):70–75, 2008.

[144] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

[145] Dae-Ki Kang, Doug Fuller, and Vasant Honavar. Learning classifiers for misuse detection using a bag of system calls representation. In *International Conference on Intelligence and Security Informatics*, pages 511–516. Springer, 2005.

[146] Kazuya Kawakami. Supervised sequence labelling with recurrent neural networks. *Ph. D. dissertation, PhD thesis. Ph. D. thesis*, 2008.

[147] Przemysław Kazienko, Edwin Lughofer, and Bogdan Trawiński. Hybrid and ensemble methods in machine learning j. ucs special issue. *J Univers Comput Sci*, 19(4):457–461, 2013.

[148] KDD-OpenSource. Anomaly generator on time series.

[149] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *null*, pages 226–233. Ieee, 2005.

[150] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, 11(1):1–27, 2007.

[151] Eamonn Keogh, Stefano Lonardi, and Bill Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 550–556. ACM, 2002.

[152] Mehdi Khashei and Mehdi Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11(2):2664–2675, 2011.

[153] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[154] JooSeuk Kim and Clayton D Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13(Sep):2529–2565, 2012.

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

[156] Serkan Kiranyaz, Turker Ince, Osama Abdeljaber, Onur Avci, and Moncef Gabbouj. 1-d convolutional neural networks for signal processing applications. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8360–8364. IEEE, 2019.

[157] Eric D Knapp and Joel Thomas Langill. *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[158] Edwin M Knorr and Raymond T Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pages 211–222, 1999.

[159] Martin Kopp, Tomáš Pevnỳ, and Martin Holena. Interpreting and clustering outliers with sapling random forests. In *Information Technologies Applications and Theory Workshops, Posters, and Tutorials (ITAT)*, 2014.

[160] Timo Koskela, Mikko Lehtokangas, Jukka Saarinen, and Kimmo Kaski. Time series prediction with multilayer perceptron, fir and elman neural networks. In *Proceedings of the World Congress on Neural Networks*, pages 491–496. INNS Press San Diego, USA, 1996.

[161] Moshe Kravchik and Asaf Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 72–83. ACM, 2018.

[162] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[163] Alp Kut and Derya Birant. Spatio-temporal outlier detection in large databases. *Journal of computing and information technology*, 14(4):291–297, 2006.

[164] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, pages 1–13, 2017.

[165] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017.

[166] Harjinder Singh Lallie, Lynsay A Shepherd, Jason RC Nurse, Arnau Erola, Gregory Epiphaniou, Carsten Maple, and Xavier Bellekens. Cyber security in the age of covid-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *arXiv preprint arXiv:2006.11929*, 2020.

[167] Dale A Lambert. A blueprint for higher-level fusion systems. *Information Fusion*, 10(1):6–24, 2009.

[168] Terran Lane and Carla E Brodley. An application of machine learning to anomaly detection. In *Proceedings of the 20th National Information Systems Security Conference*, volume 377, pages 366–380. Baltimore, USA, 1997.

[169] Terran Lane and Carla E Brodley. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49, 1997.

[170] Martin Längkvist. *Modeling time-series with deep networks*. PhD thesis, Örebro university, 2014.

[171] Martin Längkvist, Lars Karlsson, and Amy Loutfi. Sleep stage classification using unsupervised feature learning. *Advances in Artificial Neural Systems*, 2012:5, 2012.

[172] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. SIAM, 2003.

[173] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[174] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory outlier detection: A partition-and-detect framework. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 140–149. IEEE, 2008.

[175] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. In *Proc. 7th Conf. USENIX SSYM*, pages 6–20, 1998.

[176] Dan Li, Dacheng Chen, Jonathan Goh, and See-kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series. *arXiv preprint arXiv:1809.04758*, 2018.

[177] Xiaolei Li, Jiawei Han, and Sangkyum Kim. Motion-alert: automatic anomaly detection in massive moving objects. In *International Conference on Intelligence and Security Informatics*, pages 166–177. Springer, 2006.

[178] Xiaolei Li, Zhenhui Li, Jiawei Han, and Jae-Gil Lee. Temporal outlier detection in vehicle traffic data. In *IEEE International Conference on Data Engineering*, pages 1319–1322. IEEE, 2009.

[179] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[180] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[181] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 525–536, 2018.

[182] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[183] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[184] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[185] James NK Liu, Yanxing Hu, Yulin He, Pak Wai Chan, and Lucas Lai. Deep neural network modeling for big data weather forecasting. In *Information Granularity, Big Data, and Computational Intelligence*, pages 389–408. Springer, 2015.

[186] Ninghao Liu, Donghwa Shin, and Xia Hu. Contextual outlier interpretation. *arXiv preprint arXiv:1711.10589*, 2017.

[187] Chang-Tien Lu and Lily R Liang. Wavelet fuzzy classification for detecting and tracking region outliers in meteorological data. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 258–265. ACM, 2004.

[188] Huimin Lu, Yujie Li, Shenglin Mu, Dong Wang, Hyoungseop Kim, and Seiichi Serikawa. Motor anomaly detection for unmanned aerial vehicles using reinforcement learning. *IEEE internet of things journal*, 5(4):2315–2322, 2017.

[189] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[190] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[191] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[192] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.

[193] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745. IEEE, 2003.

[194] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[195] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[196] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[197] Mehrdad Mansouri, Ali Arab, Zahra Zohrevand, and Martin Ester. Heidegger: Interpretable temporal causal discovery. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1688–1696, 2020.

[198] Qian Mao, Fei Hu, and Qi Hao. Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 20(4):2595–2621, 2018.

[199] David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research*, 183(3):1466–1476, 2007.

[200] M McDonald. SHF SATCOM terminal ship-motion study. Technical report, Naval Command Control and Ocean Surveillance Center, San Diego, CA, 1993.

[201] Christoph C Michael and Anup Ghosh. Two state-based approaches to program-based anomaly detection. In *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 21–30. IEEE, 2000.

[202] Li Min and Yu Shun-Zheng. A network-wide traffic anomaly detection method based on hsmm. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1636–1640. IEEE, 2006.

[203] Roni Mittelman. Time-series modeling with undecimated fully convolutional neural networks. *arXiv preprint arXiv:1508.00317*, 2015.

[204] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018.

[205] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[206] Martin Možina, Janez Demšar, Michael Kattan, and Blaž Zupan. Nomograms for visualization of naive bayesian classifier. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 337–348. Springer, 2004.

[207] Shan Muthukrishnan, Rahul Shah, and Jeffrey Scott Vitter. Mining deviants in time series data streams. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 41–50. IEEE, 2004.

[208] Patric Nader, Paul Honeine, and Pierre Beauseroy. Detection of cyberattacks in a water distribution system using machine learning techniques. In *2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*, pages 25–30. IEEE, 2016.

[209] Alexandre Nairac, Neil Townsend, Roy Carr, Steve King, Peter Cowley, and Lionel Tarassenko. A system for the analysis of jet engine vibration data. *Integrated Computer-Aided Engineering*, 6(1):53–66, 1999.

[210] John Nicholas Newman. The theory of ship motions. In *Advances in applied mechanics*, volume 18, pages 221–283. Elsevier, 1979.

[211] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[212] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[213] Joseph L Nimmich and Dana A Goward. Maritime domain awareness: the key to maritime security. *International Law Studies*, 83(1):6, 2007.

[214] U.S. National Oceanic and Atmospheric Administration (NOAA). Presidential task force on combating illegal, unreported, and unregulated fishing and seafood fraud. `https://www.iuufishing.noaa.gov`, 2018. Accessed: August 2019.

[215] Min-hwan Oh and Garud Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1480–1490, 2019.

[216] Tomas Olsson, Daniel Gillblad, Peter Funk, and Ning Xiong. Case-based reasoning for explaining probabilistic machine learning. *International Journal of Computer Science and Information Technology*, 6(2):87–101, 2014.

[217] Cyril Onwubiko. Cyber security operations centre: Security monitoring for protecting business and supporting cyber defense strategy. In *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–10. IEEE, 2015.

[218] Shengyi Pan, Thomas Morris, and Uttam Adhikari. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Transactions on Smart Grid*, 6(6):3104–3113, 2015.

[219] Shengyi Pan, Thomas H Morris, and Uttam Adhikari. A specification-based intrusion detection framework for cyber-physical environment in electric power system. *IJ Network Security*, 17(2):174–188, 2015.

[220] Guansong Pang, Anton van den Hengel, Chunhua Shen, and Longbing Cao. Deep reinforcement learning for unknown anomaly detection. *arXiv preprint arXiv:2009.06847*, 2020.

[221] Rafał Pokrywka. Reducing false alarm rate in anomaly detection with layered filtering. In *International Conference on Computational Science*, pages 396–404. Springer, 2008.

[222] Dimitris N Politis and Halbert White. Automatic block-length selection for the dependent bootstrap. *Econometric reviews*, 23(1):53–70, 2004.

[223] David Stephen Geoffrey Pollock, Richard C Green, and Truong Nguyen. *Handbook of time series analysis, signal processing, and dynamics*. Elsevier, 1999.

[224] Leonid Portnoy. *Intrusion detection with unlabeled data using clustering*. PhD thesis, Columbia University, 2000.

[225] Yan Qiao, XW Xin, Yang Bin, and S Ge. Anomaly intrusion detection method based on hmm. *Electronics letters*, 38(13):1, 2002.

[226] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[227] Sreeraj Rajendran, Wannes Meert, Vincent Lenders, and Sofie Pollin. Saife: Unsupervised wireless spectrum anomaly detection with interpretable features. In *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–9. IEEE, 2018.

[228] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in neural information processing systems*, pages 7785–7794, 2018.

[229] Gabriëlle Ras, Marcel van Gerven, and Pim Haselager. Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 19–36. Springer, 2018.

[230] Teemu Räsänen and Mikko Kolehmainen. Feature-based clustering for electricity use time series data. In *International conference on adaptive and natural computing algorithms*, pages 401–412. Springer, 2009.

[231] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *Aaai*, volume 5, pages 1541–1546, 2005.

[232] Umaa Rebbapragada, Pavlos Protopapas, Carla E Brodley, and Charles Alcock. Finding anomalous periodic time series. *Machine learning*, 74(3):281–313, 2009.

[233] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017, 2019.

[234] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[235] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[236] Collin Rice. Idealized models, holistic distortions, and universality. *Synthese*, 195(6):2795–2819, 2018.

[237] Fred Roberts and Barry Tesman. *Applied combinatorics*. CRC Press, 2009.

[238] Pablo Romeu, Francisco Zamora-Martínez, Paloma Botella-Rocamora, and Juan Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In *International Conference on Artificial Neural Networks*, pages 451–458. Springer, 2013.

[239] Jean Roy and Steve Wark. *Concepts, models, and tools for information fusion*. Artech House, 2007.

[240] Javier Ruiz-del Solar, Patricio Loncomilla, and Naiomi Soto. A survey on deep learning methods for robot vision. *arXiv preprint arXiv:1803.10862*, 2018.

[241] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3), 1988.

[242] Ida Ruts and Peter J Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, 23(1):153–168, 1996.

[243] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[244] Christophe Salperwyck, Marc Boullé, and Vincent Lemaire. Concept drift detection using supervised bivariate grids. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2015.

[245] Stan Salvador and Philip Chan. Learning states and rules for detecting anomalies in time series. *Applied Intelligence*, 23(3):241–255, 2005.

[246] Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, 4(2), 2009.

[247] O Sami Saydjari. Engineering trustworthy systems: a principled approach to cybersecurity. *Communications of the ACM*, 62(6):63–69, 2019.

[248] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). Technical report, National Institute of Standards and Technology, 2012.

[249] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A Keim. Towards a rigorous evaluation of xai methods on time series. *arXiv preprint arXiv:1909.07082*, 2019.

[250] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536, 2015.

[251] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008.

[252] Kamran Shafi and Hussein A Abbass. An adaptive genetic-based signature learning system for intrusion detection. *Expert Systems with Applications*, 36(10):12036–12043, 2009.

[253] Amir Yaghoubi Shahir, Mohammad A Tayebi, Uwe Glässer, Tilemachos Charalampous, Zahra Zohrevand, and Hans Wehn. Mining vessel trajectories for illegal fishing detection. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1917–1927. IEEE, 2019.

[254] Thomas B Sheridan. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[255] Dominique T Shipmon, Jason M Gurevitch, Paolo M Piselli, and Stephen T Edwards. Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. *arXiv preprint arXiv:1708.03665*, 2017.

[256] G Silvestri, F Bini Verona, Mario Innocenti, and M Napolitano. Fault detection using neural networks. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 6, pages 3796–3799. IEEE, 1994.

[257] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[258] Du Sizhen, Song Guojie, Han Lei, and Hong Haikun. Temporal causal inference with time lag. *Neural Computing*, 30:271–291, 2018.

[259] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[260] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[261] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning–perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, 2005.

[262] Peter Spirtes, Clark N Glymour, Richard Scheines, David Heckerman, Christopher Meek, Gregory Cooper, and Thomas Richardson. *Causation, prediction, and search*. MIT press, 2000.

[263] Peter Spirtes and Kun Zhang. Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, page 3. SpringerOpen, 2016.

[264] SS Sreevidya. A survey on outlier detection methods. *International Journal of Computer Science and Information Technologies*, 5(6):8153–8156, 2014.

[265] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2828–2837, 2019.

[266] Pei Sun, Sanjay Chawla, and Bavani Arunasalam. Mining for outliers in sequential databases. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 94–105. SIAM, 2006.

[267] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[268] Boleslaw K Szymanski and Yongqiang Zhang. Recursive data mining for masquerade detection and author identification. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 424–431. IEEE, 2004.

[269] Xiaobin Tan and Hongsheng Xi. Hidden semi-markov model for anomaly detection. *Applied Mathematics and Computation*, 205(2):562–567, 2008.

[270] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. Attack models and scenarios for networked control systems. In *Proceedings of the 1st international conference on High Confidence Networked Systems*, pages 55–64. ACM, 2012.

[271] Pooja Thakkar, Jay Vala, and Vishal Prajapati. Survey on outlier detection in data stream. *Int. J. Comput. Appl*, 136:13–16, 2016.

[272] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): towards medical xai. *arXiv preprint arXiv:1907.07374*, 2019.

[273] Geoffrey G Towell and Jude W Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine learning*, 13(1):71–101, 1993.

[274] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125, 2016.

[275] Thomas J Veasey and Stephen J Dodson. Anomaly detection in application performance monitoring data. In *Proceedings of International Conference on Machine Learning and Computing (ICMLC)*, pages 120–126, 2014.

[276] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[277] Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364, 2006.

[278] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, volume 8, page 6, 2013.

[279] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Learning reliable visual saliency for model explanations. *IEEE Transactions on Multimedia*, 2019.

[280] Zhiguang Wang and Tim Oates. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, volume 1, 2015.

[281] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

[282] Li Wei, Nitin Kumar, Venkata Nishanth Lolla, Eamonn J Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. Assumption-free anomaly detection in time series. In *SSDBM*, volume 5, pages 237–242, 2005.

[283] Andrew W Williams, Soila M Pertet, and Priya Narasimhan. Tiresias: Black-box failure prediction in distributed systems. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[284] Ian H Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77, 2002.

[285] Elizabeth Wu, Wei Liu, and Sanjay Chawla. Spatio-temporal outlier detection in precipitation data. In *Knowledge discovery from sensor data*, pages 115–133. Springer, 2010.

[286] Xin Xu. Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies. *Applied Soft Computing*, 10(3):859–867, 2010.

[287] Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

[288] Nong Ye. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169. West Point, NY, 2000.

[289] Yimin Ye and Nicolas L Gutierrez. Ending fishery overexploitation by expanding from local successes to globalized solutions. *Nature Ecology & Evolution*, 1(7):0179, 2017.

[290] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.

[291] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 2019.

[292] Wojciech Zamojski, Jacek Mazurkiewicz, Jarosław Sugier, Tomasz Walkowiak, and Janusz Kacprzyk. Theory and engineering of complex systems and dependability. In *Proceedings of the Tenth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX*. Springer, 2015.

[293] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[294] Xiaoqiang Zhang, Pingzhi Fan, and Zhongliang Zhu. A new anomaly detection method based on hierarchical hmm. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 249–252. IEEE, 2003.

[295] Qingyu Zhao, Ehsan Adeli, Nicolas Honnorat, Tuo Leng, and Kilian M Pohl. Variational autoencoder for regression: Application to brain aging analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 823–831. Springer, 2019.

[296] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[297] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

[298] Zahra Zohrevand and Uwe Glässer. Dynamic attack scoring using distributed local detectors. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[299] Zahra Zohrevand and Uwe Glässer. Should I Raise The Red Flag? A comprehensive survey of anomaly scoring methods toward mitigating false alarms. *arXiv preprint arXiv:1904.06646*, 2019.

[300] Zahra Zohrevand, Uwe Glässer, Hamed Yaghoubi Shahir, Mohammad A Tayebi, and Robert Costanzo. Hidden markov based anomaly detection for water supply systems. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1551–1560. IEEE, 2016.

[301] Zahra Zohrevand, Uwe Glässer, Mohammad A Tayebi, Hamed Yaghoubi Shahir, Mehdi Shirmaleki, and Amir Yaghoubi Shahir. Deep learning based forecasting of critical infrastructure data. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1129–1138. ACM, 2017.

[302] Yijun Zuo and Robert Serfling. General notions of statistical depth function. *Annals of statistics*, pages 461–482, 2000.