Unsupervised Temporal Anomaly Detection

Time series, Data stream, and Interpretability

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Bin Li

Dortmund
2024

Tag der mündlichen Prüfung: 11.07.2024

Dekan: Prof. Dr.-Ing. Gernot Fink
Gutachter: Prof. Dr. Emmanuel Müller, Prof. Dr. João Gama

Abstract

Anomaly detection becomes essential across diverse domains. Data is usually collected sequentially in real-world applications such as sensor records and network logs. Consequently, a major challenge in anomaly detection is the real-time volatile sequential abnormal events. Recent research on time series has gained supreme advancements, leveraging the vast development of deep models like recurrent neural networks and transformers. However, most existing deep models focus on static time series while neglecting the dynamic streaming feature inherent in real-world deployment. A critical issue arises from the potential occurrence of distributional drift in streaming data, after which the pre-trained models become invalid. Furthermore, as machine learning models are applied in the safety-crucial fields like autonomous vehicles and medical diagnoses, the trustworthiness of model predictions becomes a growing concern. A desired anomaly detector is expected to both predict and interpret the abnormal events.

This dissertation focuses on the intersecting research area between time series and data stream anomaly detection as well as their interpretability. We first develop a contrastive-learning-based self-supervised approach for time series anomaly detection, contributing to the effective representation learning of time series anomalies without labels. Subsequently, we investigate a novel concept drift detection approach for identifying correlation changes in the data stream. We also propose a state-transition-aware online anomaly detection framework for data streams. Finally, we delve into the necessary properties of time series interpreters, including cohesiveness, consistency, and robustness. We also showcase an example-based interpreter for reconstruction-based anomaly detection models, which provides intuitive and contrastive explanations of the reasons behind anomalies. The proposed approaches are rigorously evaluated on various popular real-world benchmark datasets and simulations.

Publications

This dissertation is based on the following publications. All works are supervised by Emmanuel Müller. [BLM23a] and [BLM23b] are collaborated with Chiara Balestra with equal contributions. Carsten Jentsch serves as supervisor and editor for [LJM22].

[BLM23a] Chiara Balestra, Bin Li, and Emmanuel Müller. On the consistency and robustness of saliency explanations for time series classification. In *The 9th SIGKDD International Workshop on Mining and Learning from Time Series (MILETS 2023)*, 2023.

[BLM23b] Chiara Balestra, Bin Li, and Emmanuel Müller. slidshaps–sliding shapley values for correlation-based change detection in time series. In *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2023.

[LJM22] Bin Li, Carsten Jentsch, and Emmanuel Müller. Prototypes as explanation for time series anomaly detection. *The 2nd KDD Workshop on Anomaly and Novelty Detection, Explanation and Accommodation (ANDEA'22)*, 2022.

[LM23a] Bin Li and Emmanuel Müller. Contrastive time series anomaly detection by temporal transformations. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023.

[LM23b] Bin Li and Emmanuel Müller. State-transition-aware anomaly detection under concept drifts. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 49–63. Springer, 2023.

[LM24] Bin Li and Emmanuel Müller. Cohesive explanation for time series prediction. In *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024.

PUBLICATIONS

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Part I

Introduction

1

Introduction

"An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism" [Haw80]. Hawkins' definition from the 1980s has been commonly used as a general definition of outliers. In recent decades, outliers have gained their unique definition in more and more machine learning application areas. Accordingly, the outlier detection approaches have also evolved from statistical and density-based methods to recent deep-learning models. Despite the data becoming more complex and the model becoming more capable, the essential implication of outliers stays the same, i.e., data points deviating from the studied mechanism or distribution. Under the common "independent and identically distributed" (i.i.d.) assumption in classical machine learning, outliers show deviating behaviors, e.g., machine failure, physical disease, and unexpected traffic scenarios. Although noise refers also to deviating data points in the machine learning context, they are usually less informative or interesting to the user. Certain prepossessing techniques usually eliminate noise before the machine learning tasks. However, outliers are usually worth investigating and bring potential valuable knowledge to the system.

Outlier detection has been researched as an essential task in both academia and industry. While the term "outlier" usually refers to a statistically extreme value within the dataset, in

the industry, "anomaly" refers to broader extreme events in the general application context. Although "outlier" and "anomaly" are interchangeably used in several scenarios, in this work, we stick to "anomaly" targeting a more applied research domain. Correspondingly, the opposite of "anomaly" is referred to as "normal" [1].

With the rapid industrial digitization in the last decade, the data format in machine learning tasks has evolved from tabular data to more complex formats such as text, image, and video. Previously, anomalies typically were outlying points in the relational database system, some of which are even detectable by setting a single threshold. Detecting anomalies becomes challenging due to the intricate representation of recent data. In this work, we focus on detecting anomalies in *temporal data*, a series of sequential data collected in time order. Specifically, we distinguish temporal data between **time series** and **data streams**. With time series, we refer to the whole stationary sequence that is available for machine learning processing. We are detecting abnormal events during certain periods in the given sequence. In contrast, a non-stationary data stream is closer to real-world applications. Data arrives continuously as a stream from certain data sources, e.g., sensor networks and wearable devices. Due to the nature of dynamic data streams, distributional drifts are one of the major challenges in modeling streaming data. We target detecting anomaly events in dynamic data streams given the latest temporal context of the data. We stick to the unsupervised paradigm because of the expensive label acquisition in the real-time environment.

Recently, deep models have been developed for various challenging tasks, including temporal anomaly detection. Although proficient deep models have achieved significant success, there is an increasing concern about the trustworthiness of these black-box models. Especially in safety-crucial applications, it is vital to understand the reason for machine learning model predictions. Specifically in anomaly detection, users expect not only a precise alarm of abnormal events but also a proper reasoning of the cause.

To this end, in this dissertation, we research the intersecting area between time series and data stream anomaly detection and their model interpretability. In the rest of this chapter, we first introduce the common applications in this area and existing solutions in Section 1.1. Furthermore, we emphasize the major challenges in Section 1.2. Finally, we summarize the contribution in the remaining chapters in Section 1.3.

[1] Unless specifically stated, the term "normal" only refers to the opposite of "abnormal" in the anomaly detection context, instead of normal distribution.

1.1 Background

1.1.1 Applications

Time series anomaly detection (TSAD) is an essential task in various application branches, many of which require streaming anomaly detection due to the real-time demand. In recent years, the vast development of IoT techniques has enabled rapid industrial digitization. Anomaly detection in sensor networks is an essential subtask in IoT systems. For example, Mathur et al. [MT16] detect time series anomalies from a water treatment testbed in the industrial control system. By analyzing the sensor data, they aim to detect simulated attacks on the testbed devices (e.g., water tank overflow). Similarly, HAI [SLYK20] is collected from another industrial testbed for steam turbine power and pumped-storage hydropower generation. By collecting sequential sensor data, they also aim to detect simulated abnormal attack scenarios. Predictive maintenance is another important industrial application domain of TSAD. Machine failure can potentially lead to expensive consequences. Early detection of abnormal machine behaviors can prevent or reduce repair costs [CLAM+21, dSAdSAF+21, KS20]. TSAD also plays a vital role in IT security [SGF+02, TJ03, DLZS17]. Network and system attacks are evolving and persistent challenges in the IT security domain. Network traffic and access logs contain temporal consistent information, which can be used to indicate potential attacks. Compared to traditional signature-based intrusion detection systems [IVML18], modern approaches [SZ18] detect anomalies considering the temporal network access behavior. In the finance domain, credit card fraud can be detected by analyzing the consumption history. In environment monitoring, evolving environment data (e.g., humidity, air temperature, wind velocity) has been used to predict forest fire risk in Australia [SR18].

For some safety-crucial domains, trustworthiness and reliability become increasingly important factors in addition to anomaly detection performance. For instance, detecting abnormal physical conditions in the medical diagnosis domain requires reliable machine learning models and explainable predictions, e.g., detecting arrhythmia in ECG data [CF07, PS19, CV15]. Recent wearable devices also support automatic emergency calls for either physical diseases or accidents. TSAD also contributes to the aerospace field [HCL+18]. For instance, trajectory anomaly detection of spacecraft helps with real-time monitoring. However, untrustworthy predictions and decisions may lead to catastrophic and irreparable consequences. Finally, anomaly detection is also important in autonomous vehicles [BNZ22]. Application spreads across multiple sensor modalities (e.g., camera, lidar, radar).

1.1.2 Common anomaly detection approaches

Anomaly detection has been widely studied as an essential machine-learning task. Solution frameworks spread from classical machine learning to recent deep learning models. The recent solutions are well summarized in the literature [HHH+22]. Here, we give an overview of the popular approaches from the supervision perspective. We extensively go through the unsupervised approaches most relevant to our work while summarizing the ideas of supervised and semi-supervised approaches from a broad view.

1.1.2.1 Unsupervised approaches

Due to the biased distribution between normal and abnormal data, unsupervised approaches are considered more practical for real-world problems. A common assumption in unsupervised anomaly detection solutions is that pure normal data is available in the training phase [LZX+21]. The task is to detect abnormal samples in the test phase without label information.

In **classical machine learning**, various categories of approaches solve the anomaly detection task in an unsupervised fashion. Statistical approaches work under certain probability distribution assumptions. Normal points are supposed to lie in the probability-dense regions, while anomalies can be observed with less probability. For example, according to the *empirical rule*, instances more than two or three standard deviations away from the mean in a Gaussian distribution can be considered anomalies. In the case of multivariate data, the statistical approach can be extended to measure the Mahalanobis distance between an object and the multivariate Gaussian distribution parameterized by the mean and covariance matrix. Obviously, these approaches work under the strong assumption of the data distribution and the cut-off parameter.

Distance-based approaches belong to another category of solutions. An object o is defined as a $DB(minPts, \epsilon) - outlier$ if there are less than $minPts$ objects within the neighborhood area determined by radius ϵ. This approach is limited in large datasets due to the nested loop of database scan with $O(n^2)$ complexity. Furthermore, a single pair of parameters ($minPts$ and ϵ) can also not capture datasets with different densities.

Targeting the limitation of Distance-based approaches, the density-based approaches further model the neighborhood density definition with more advances. LOF [BKNS00] detects outliers by accessing the local densities of instances. Kernel density estimation extends LOF for high-dimensional applications [SZK14, LLP07].

Reconstruction-based approaches serve as unsupervised anomaly detectors with the basic idea of learning shared patterns of normal data by reconstructing the instances. Instances with poor reconstruction in the test phase are suspected to be abnormal. A linear model from this category is to reconstruct instances by non-negative matrix factorizations [AGAA18]. The input matrix is mapped into the latent space and then reconstructed using two low-ranked matrices, which can be acquired from, for instance, singular value decomposition (SVD).

Other variants of classical machine learning models are also designed for anomaly detection in an unsupervised fashion. Isolation Forest [LTZ08] extends the supervised tree-based random forest model for anomaly detection by "isolating" outlier points using the tree branches. Outlier points are supposed to be contained in shorter branches in the tree. OCSVM [SWS+99] extends the classical Support Vector Machines. Instead of maximizing the margin between classes, OCSVM is trained with only normal data. It seeks to find a hyperplane that separates the normal instances from the origin in the feature space. This hyperplane is chosen to maximize the margin between the hyperplane and the nearest normal instances. Similarly, SVDD [TD04] minimizes the volume of the hypersphere around the normal data.

Despite the efficiency and efficacy of classical machine learning methods, they usually fail on complex (e.g., high-dimensional, temporal) data. Various **deep models** have recently been developed for anomaly detection in complex data. Similar to classical machine learning, deep reconstruction-based approaches are popular in unsupervised anomaly detection. Autoencoders are designed to reconstruct the input data, and the reconstruction error of unseen data indicates the likelihood of being an anomaly. The application area of Autoencoder-based anomaly detection spread across time series data [SY14], data streams [CK22], image data [PSvdH19], video scenes [XRY+15], etc. Different than minimizing the reconstruction error, DeepSVDD [RVG+18] minimizes the hypersphere volume enclosing the normal data represented by a deep neural network.

1.1.2.2 Supervised approaches

Anomaly detection can be treated as a supervised task for classifying samples into normal and anomaly classes. However, classical classifiers (e.g., random forest [Ho95], support vector machine [CV95]) may fail in this setting due to the biased class sample distribution. In anomaly detection, the majority of samples are usually from the normal class, while anomalies are supposed to appear only rarely. Therefore, balancing the two classes by resampling is a necessary pre-processing step to use classical supervised models.

Another critical issue is the unseen anomalies [HHH+22]. Although normal data is usually fully represented in the train set, anomalies can be any unknown patterns that differ from the normal one. Therefore, a binary classifier can only be trained to distinguish existing patterns in the train set, which causes large uncertainty on the unknown anomalies in the test phase.

1.1.2.3 Semi-supervised approaches

Semi-supervised approaches assume partial label availability. They use the available labels to learn representations of the normal data while keeping flexible detection ability of unseen anomalies. Some representative approaches are DeepSAD [RVG+19], an extension of the unsupervised DeepSVDD [RVG+18] method that uses labels to enforce anomalies being represented away from the center of normal data; DevNet [PSvdH19] uses a few labeled anomalies to enforce deviation of the anomaly scores of anomalies from normal objects in an end-to-end neural deviation learning.

1.1.3 Advanced settings

Despite the vast development of anomaly detection techniques, their deployment in real-world scenarios still faces vital challenges. In the era of big data, machine learning paradigms require advanced adaptation for complex data formats. Here, we discuss the advanced anomaly detection methods in streaming and high-dimensional data.

1.1.3.1 Streaming anomaly detection

Anomaly detection in data streams is a common application in many real-world domains, e.g., predictive maintenance [CLAM+21, dSAdSAF+21, KS20], medical diagnosis [CF07, PS19, CV15], telecommunication [FRC+19]. The input data is generated in real-time, while the challenging task is to detect abnormal events in an online streaming fashion, where the labeling is expensive and the processing time is limited. One common strategy is applying sliding windows over the arriving data stream and employing static unsupervised approaches in Section 1.1.2.1 to the recent buffered data. Streaming clustering algorithms [PZX+18, CEQZ06, APHW03] can be used for online anomaly detection, where tiny or single-element clusters are the targets. Some static anomaly detection approaches have also been adapted for the streaming setting, e.g., incremental LOF [PLL07], stream-adapted Isolation Forest [DF13].

Dedicated streaming anomaly detection approaches have also been developed for this purpose. xStream [MLA18] is a density-based approach that detects abnormal streaming data at different granularities. LODA [Pev16] is an online ensemble approach that combines lightweight weak histogram-based anomaly detectors for online processing. HS-tree [TTL11] is a tree-based online anomaly detector with low complexity enabled by efficient tree splitting.

Considering the limited capability of shallow models on complex streaming data, deep models have also been developed for this target. HTM [ALPA17] is a neural network model trained unsupervised and dynamically adapted to the latest statistics of data streams. Similar to the ensemble shallow methods, deep Autoencoders have also been aggregated by pooling strategies [YLLL22] for online anomaly detection.

1.1.3.2 Subspace anomaly detection

In addition to the high-velocity and volatile streaming data, the high dimensionality is another major challenge when deploying anomaly detection models for real-world applications. *Curse of dimensionality* describes a failure effect of classical machine learning models in high-dimensional space. Classical distance measurements become meaningless, and the algorithm scalability reduces drastically. Detecting anomalies in high-dimensional data requires efficiently identifying the relevant features (i.e., the subspace) where the abnormal events occur. HiCS [KMB12] is a density-based approach that ranks outlier scores in high contrast (relevant) subspaces. SOD [KKSZ09] infers relevant attributes from the neighbors of each data point.

Deep models are naturally more capable of handling high-dimensional data than shallow models, thanks to their deep structures and quantity of trainable parameters. In addition to the common deep models mentioned in Section 1.1.2, more and more deep neural networks are proposed for the anomaly detection task. Recent advancements include deep generative models [HCL21, ZFL+18, ZLH+19], diffusion models [XGT+23, LZWW23, LJHR23], transformers [XWWL21, TCJ22, YZZ+23] and large language models [GZZ+23, ZPT+23, HYT23].

1.1.4 Anomaly interpretation

An essential target of anomaly detection is to analyze the causes and prevent further appearance or take early treatments (e.g., machine failures, disease diagnosis). Therefore, the interpretation of abnormal data and the decisions of anomaly detectors are vital. Interpreting black-box machine learning models is still understudied but has recently become popular.

Interpretation approaches are commonly categorized by various criteria. *Global* approaches provide insights into overall model behavior across the entire dataset, while *local* approaches focus on explaining an individual instance or prediction. *Model-specific* interpretation are designed for specific model structures, utilizing their internal characteristic, while *model-agnostic* interpretation can be applied to universal models without prior knowledge of their internal structure. *Intrinsic* approaches enable interpretability inherently from the model design and learning process, while *post-hoc* approaches are applied after the model training. Targeting temporal anomaly detection in long-lasting and high-dimensional time series, in this dissertation, we focus on **local** interpretations, providing more clear explanations to individual anomaly cases, **model-agnostic** and **post-hoc** approaches being more flexible to anomaly detectors.

In the existing interpretable machine learning approaches, some general-purpose approaches can be applied to anomaly detection tasks [TCD+22, IGCBF20]. Other than them, dedicated approaches are designed with more careful consideration of the challenges in anomaly detection tasks; some can also complete the interpretation together with anomaly detection in an end-to-end fashion. ACE [SSFE23] provides unsupervised concept-based binary interpretation. It explains each prediction by indicating whether a human-interpretable concept is in the target input. Mask-based approaches [LMA23] are commonly used to analyze the attributions of features to the abnormal event. Singh et al. [SJLM23] generate task-specific explanations for video anomaly localization. In this work, we will investigate the necessary properties of time series interpreters, focusing on anomaly detection.

1.1.5 Resources and limitations

Current time series and data stream anomaly detection approaches are usually compared and evaluated based on common benchmark datasets and library implementations. In this section, we summarize a few popular existing resources in this field and discuss their limitations and potential impact on new anomaly detectors.

1.1.5.1 Existing resources

Textbooks and surveys: Classical machine learning models to our research topics are well described in various textbooks, including anomaly detection [Agg16], data stream analysis [Gam10] and interpretable machine learning [Mol22]. Recent advances are well summarized in the following surveys. Choi et al. [CYPY21] and Darban et al. [DWP+22] summarize the state-of-the-art TSAD approaches in their works. Lu et al. [LLD+18] systematically review the drift

detection and modeling approaches. And Li et al. [LZVL23] provide an overview of the explainable anomaly detection techniques.

Benchmark studies: A few benchmark studies have covered the performance evaluation of a vast amount of popular anomaly detectors. From a theoretical perspective, Ruff et al. [RKV+21] benchmark various deep and shallow anomaly detection models. ADBench [HHH+22] conducts extensive experimental comparisons among the latest models. Lai et al. [LZX+21] conduct a TSAD benchmark study. Ismail et al. [IGCBF20] conduct a benchmark study on time series interpretation approaches.

Libraries: Existing anomaly detection models are well-implemented by various open-source libraries. PyOD [ZNL19], RapidMiner [AG12], River [MHM+21] and Weka [HFH+09] are designed for general purpose, TODS [LZW+21] and DeepADoTS [1] are for TSAD, PySAD [YK20] and MOA [BHKP10] support streaming data anomaly detection, and Captum [KMM+20] implements a batch of interpretability models.

Datasets: The following repositories well cover standard benchmark datasets for anomaly detection. ODDS [Ray16] and UCI [AN07] contain datasets for general purpose, UCR [CKH+15] is a recent repository with TSAD datasets. Besides public data archives, synthetic data are also often used for streaming model evaluation. Popular ones include scikit-multiflow [MRBA18] and agots [2].

1.1.5.2 Limitations

Despite the resources in the TSAD domain, the approaches and the benchmark datasets still have some limitations. Firstly, anomaly detectors are usually designed and evaluated in a task-specific fashion, i.e., telemetry sensors [HCL+18], water treatment [DH21, HW22]. A significant consequence is that those approaches are not cross-validated by other benchmark datasets. In addition, we also observe that TSAD and data stream concept drift detection are usually researched separately. Joint research on distinguishing anomalies and drifts, as well as adaptively updating anomaly detectors with data streams, are still understudied.

From the data perspective, though several popular benchmark datasets are employed in the evaluation of multiple approaches (e.g., SMD [SZN+19], SMAP and MSL [HCL+18]), Keogh et al. [WK21] point out that they are flawed. Even a single line of MATLAB code can detect many trivial anomalies in these popular datasets. In contrast, the UCR [CKH+15] archive is

[1] https://github.com/KDD-OpenSource/DeepADoTS
[2] https://github.com/KDD-OpenSource/agots

more reliable. However, a well-qualified multivariate TSAD benchmark dataset is still rare to see. Given the severe situation, we claim that anomaly interpretation is an efficient way of understanding anomalies and identifying unreliable predictions of biasedly evaluated anomaly detectors.

1.2 Challenges

As discussed in Section 1.1, various approaches and resources have been investigated in the anomaly detection and interpretation domains, given that there are still considerable flaws in the evaluation procedure. In this section, we highlight the existing challenges in the studied intersecting research area. An overview is depicted in Figure 1.1. Generally, detecting and explaining time series and streaming anomalies suffer under complex temporal information, dynamic data characteristics, as well as black-box predictions.

1.2.1 Static time series

Similar to anomaly detection in other domains (e.g., tabular, image, text data), TSAD usually faces a lack of labeled data. On the one hand, labeled anomalies by domain experts is often expensive. On the other hand, anomalies only appear rarely compared to normal data. Therefore, supervised approaches often under-represent the anomaly class. Although the normal class can usually be represented by a limited amount of patterns, anomalies are endless, i.e., any pattern differing from the predefined normal pattern can be considered an anomaly.

Specifically in TSAD, the contextual dependency on time series data makes the severe problem even more challenging. One significant difference between time series and tabular data is the interdependent relationship between instances at adjacent timestamps. Hence, the patterns learned from time series should be temporal patterns describing certain periods. The windowing technique is one of the most commonly used strategies to collect temporal instances, e.g., landmark windows, damped windows, and sliding windows [WC06]. One significant challenge here is determining a proper window size, which should be large enough to cover the necessary temporal context yet not redundant.

The temporal-dependent features also play a vital role in defining time series anomalies. *Point anomalies* are similar to anomalies in the non-temporal context, where a data instance at a single timestamp is outlying from the global data distribution. Additionally, in TSAD, the *contextual anomalies* are more of concern, which are abnormal events consisting of multiple

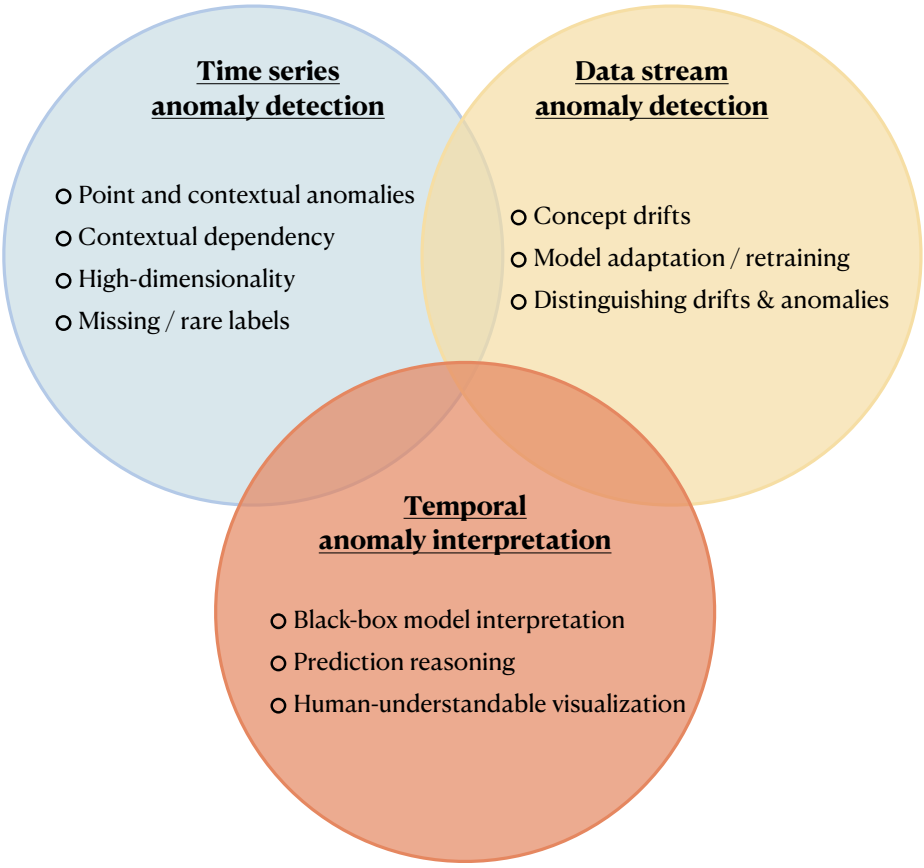


Figure 1.1: Challenges in the intersecting research area. The challenges correspond to the proposed solution set: Part II focuses on time series anomaly detection; Part III on data stream anomaly detection; Part IV on temporal anomaly interpretation.

temporally adjacent data instances in a period. Detecting contextual anomalies is challenging while every single instance within the abnormal event period might still show normal behavior w.r.t. the point-wise anomaly definition, however, the holistic pattern is outlying.

1.2.2 Dynamic data streams

In data stream anomaly detection, temporal information and contextual anomalies are major challenges. In the dynamic streaming environment, *concept drifts* become unavoidable. In classical machine learning tasks, including TSAD, we usually make the i.i.d. assumption. Namely, we assume the data for both training and testing are uniformly and independently sampled from an identical distribution. However, in data streams, the data distribution can also change over time with the evolving environment. This could make the model initialized on one

data distribution no longer valid after a distributional change, i.e., concept drift. In the context of anomaly detection, concept drifts raise the issue: normal data defined in one distribution may become anomalies after a concept drift, and vice versa. Hence, an efficiently adaptive anomaly detector is desired for the streaming environment.

In addition, both abnormal events and concept drifts indicate receiving newly arriving data instances different from the last observations. One crucial task is to distinguish anomalies from concept drifts. Although concept drifts can also be considered as anomalies w.r.t. to the previous data distribution, in practice, they are usually less interesting to be defined as anomalies. After observing concept drifts, the primary task is to trigger the model adaptation and update the model to the latest distribution.

1.2.3 Interpretability

In addition to prediction performance, interpretability has been increasingly considered one of the most critical aspects of machine learning models, especially in safety-crucial applications. A desired anomaly detection model should not only precisely detect the abnormal events but also provide proper reasoning of the cause. Interpreting anomalies can help users understand the abnormal event early, apply proper treatment, and possibly prevent future recurrence. Existing interpretability approaches are usually not directly applicable to TSAD models. Interpretation models designed for general purposes do not focus on the temporal information in time series. A proper explanation of time series prediction should based on the temporal event that causes the prediction. Due to the complex temporal (timestamps) and spatial (features) characteristics of time series, visually understanding the raw input data is already challenging for humans. Therefore, the interpretation of time series anomalies should be efficient and simplified. Another challenge is interpreting anomalies. In order to understand the type and cause of anomalies, it is usually necessary to first understand what is normal. Although classical feature attribution approaches help to figure out the most contributing features and timestamps, they still need intuitive interpretations. Instead, a *contrastive interpretation* between normal and abnormal is informative for humans.

In Figure 1.2, the three research topics of this dissertation are visualized on a problem axis. From time series anomaly detection via data stream anomaly detection to temporal anomaly interpretation, the hardness increases due to the problem setting and the task changes from theoretical and performance-oriented (e.g., optimizing detection accuracy) to practical and human-oriented (e.g., industrial deployment and interpretation).

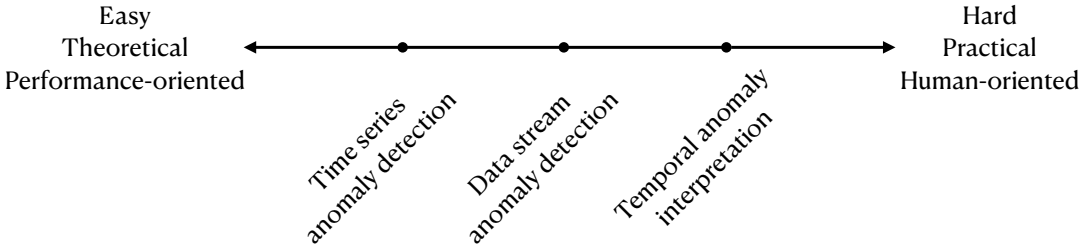


Figure 1.2: Problem axis.

1.3 Contributions

In this dissertation, we study the intersecting research area between time series and data stream anomaly detection, as well as their interpretation. An overview of the covered topics and the corresponding publications are listed in Table 1.1. We first develop a general approach for TSAD focusing on the temporal features in Part II. In Part III, we extend the problem setting to real-time data streams. Specifically, we study the impact of concept drift on static TSAD models and possible model adaptation strategies. Finally, aiming at the practical deployment in real-world applications, especially targeting safety-crucial domains, we investigate interpretation approaches on time series prediction and anomaly detection in Part IV.

In Chapter 2, we develop a self-supervised TSAD model ContrastAD [LM23a]. Inspired by the convincing performance of contrastive learning in image data representation learning, we extend its benefit of representing the normal class without accessing anomaly samples in the TSAD domain. Facing the challenging construction of contrastive pairs for time series data, we design anomaly-induced temporal transformation.

In the data stream setting in Part III, we firstly revisit the drift detection techniques in Chapter 3. Anomalies in the data stream are defined in the context of their temporal context. Therefore, drift detection techniques contribute to identifying the timestamps or periods where the hidden data distribution in the stream drifts. Given the expensive labeling process in real-time, different from common drift detectors [GMCR04, BG07], we focus on the prior distribution changes (i.e., virtual concept drifts [LLD+18]) in the temporal anomaly detection context. We propose slidSHAPs [BLM23b], an unsupervised drift detection approach based on the feature correlation drifts derived from Shapley values [Sha53]. After that, in Chapter 4, we delve into the anomaly detection task in drifting data streams. We propose STAD [LM23b], a state transition model that captures the anomaly detector performance w.r.t. the drifting data distributions

Table 1.1: Topics overview

	Topic	Publication
Part I	Chapter 1 Introduction	
Part II	Chapter 2 Contrastive time series anomaly detection	[LM23a]
Part III	Chapter 3 Correlation drift detection	[BLM23b]
	Chapter 4 Online adaptive anomaly detection	[LM23b]
Part IV	Chapter 5 Cohesive time series explanation	[LM24]
	Chapter 6 Consistency and robustness	[BLM23a]
	Chapter 7 Example-based explanation	[LJM22]
Part V	Chapter 8 Conclusion & future work	

in an online fashion. We design automated real-time model selection and transition paradigms that optimize the anomaly detector to the latest data patterns.

While the approaches proposed so far optimize for the anomaly detection performance in both static and dynamic settings, in Part IV, we study the interpretability of the temporal anomaly detectors, focusing on *local*, *model-agnostic* and *post-hoc* interpretations. We also define the essential demand of interpreting time series predictions. In Chapter 5, we emphasize the challenge of using saliency maps to interpret the complex and less human-readable (compared to image or text data) time series data. We propose CETS [LM24], a two-stage cohesive time series prediction interpretation approach, which simplifies the attribution analysis of both sub-feature space and time period of each input data window. Furthermore, for future research, in Chapter 6, we point out the existing issue that the attribution consistency between adjacent time windows is not guaranteed by classical machine learning interpretation approaches. Similar to the time dimension, in the feature dimension, we also demonstrate that the time series interpretation approaches require robustness against feature permutation. Subsequently, in Chapter 7, we provide a prototypical study on interpreting time series anomaly in practice. We propose ProtoAD [LJM22], an example-based interpretation approach for reconstruction-based anomaly detectors. Visualizing examples of normal and abnormal samples intuitively shows the reasoning behind anomalies.

As a conclusion of this dissertation, in Part V, we summarize our contributions to the research area and sketch potential further research gaps for future works.

Part II

Time Series Anomaly Detection

2

Contrastive Time Series Anomaly Detection

The complex and volatile temporal features make temporal anomaly detection in time series and data streams extremely challenging. In this chapter, we start with anomaly detection in static time series, where sufficient data is available for the training, and the dataset is uniformly sampled. We address the challenges in Section 1.2.1, specifically, in time series data, some anomalies only show deviating patterns to their local temporal context instead of the overall distribution. Furthermore, the biased sample distribution between normal and abnormal classes hinders the efficient usage of the labels. Self-supervised approaches are practically efficient for anomaly detection, in which only normal data is used during the training. However, they often fail to detect contextual anomalies in high-dimensional time series data, while the representation learning of such temporal data patterns is sub-optimal.

In this chapter, we propose ContrastAD, a novel self-supervised framework for time series anomaly detection (TSAD). Specifically, we employ the contrastive learning process with anomaly-induced temporal transformations. Targeting the point and contextual anomalies in time series data, we develop corresponding transformations to enforce the model to learn dis-

crepant representations for normal and abnormal data in the latent space. With extensive experiments, we show that our approach outperforms baseline anomaly detectors on various benchmark datasets. Our empirical results indicate that ContrastAD improves anomaly detection performance on noisy and high-dimensional time series datasets.

2.1 Introduction

Anomaly detection in time series data has attracted vast attention in recent machine learning research. Classical database anomaly detection approaches [LTZ08, BKNS00, EKS+96, SWSST00] usually work efficiently on low-dimensional data and detecting *point anomalies*. Meanwhile, temporal information is a major aspect when considering *contextual anomalies* in time series. Time series forecasting-based models can detect anomalies without losing historical information [RL05]. Reconstruction-based models [Hof07] belong to another thread of approaches that use the reconstruction error to indicate the likelihood of anomaly without accessing labels. However, those models often fail when the complexity and dimensionality of the time series increase. A sub-optimal representation of the data hinders anomaly detection tasks.

Recently, multiple deep models have achieved promising performance on standard benchmark datasets. The deep models capture long-term temporal information in the time series sequences with their deep structures. Learning informative representation with deep models facilitates anomaly detection enormously. Recurrent Neural Networks (RNNs) are used in both forecasting-based [MVS+15] and reconstruction-based models [MRA+16, SY14]. The recurrent units are supposed to aggregate important information from history. Temporal Convolutional Networks (TCNs) model the temporal data with convolution kernels [BKK18]. Transformers are also applied to time series data where the attention mechanism extracts temporal dependencies between timestamps from the time series data [XWWL21, TCJ22, ZJP+21]. Despite the powerful modeling capacity of the deep models, they usually focus on intra-instance information (e.g., within a sliding window) while neglecting the semantic relation between instances. Inter-instance information (between sliding windows) is especially important for contextual dependency analysis between instances.

Contrastive learning (CL) is a self-supervised learning paradigm that learns underlying representation from unlabeled data [JBZ+20]. It has achieved notable breakthroughs in multiple

computer vision tasks [CSL21, SLY+21, GEY18]. The main idea is to augment the original data with multiple transformations and discriminatively learn to distinguish between them. Previous works commonly follow the strategy of augmenting the input data with different transformations, then deriving *positive* and *negative* pairs from the augmented data. The learning objective encourages relevant data (*positive pairs*) to stay together in the latent space while irrelevant data (*negative pairs*) to be apart from each other. This procedure allows the model to learn to differentiate features between instances. We are motivated to apply CL to TSAD tasks to enable the model to recognize the imperceptible contextual anomalies in time series. In the CL-based anomaly detection models, data transformations facilitate learning useful features for detecting novelties [GEY18]. A downstream auxiliary classification task is often used to predict the anomaly score [TMJS20, SLY+21, WBR+20, SCM21]. It has been shown that the contrastive loss can also be used directly as the anomaly score if no label is available to train a classifier [SW22, QPK+22]. The anomaly data are supposed to cause larger contrastive loss due to sub-optimally distinguishing the original and transformed data.

CL has already been employed in many image [WBR+20, SLY+21, TMJS20] but few in time series [QPK+22, SQK+22] anomaly detection tasks. Thanks to the nature of image data, it is straightforward to apply transformations to the raw data and further define contrastive pairs for the learning process. Common transformations include geometric transformations (e.g., rotation, flipping, reflection) [GEY18] and jittering (i.e., adding random noise) [WLM+22]. CL has yet to be widely developed for time series data while finding informative transformation for temporal data is not trivial. Existing works apply an autoregressive model for latent space forecasting [OLV18] or learn transformations by a dedicated neural network [QPK+22]. However, they are not designed for TSAD and explicitly target representing the normal patterns as well as the highly contrasting point and contextual anomalies. The contribution of such transformations to time series data with context-dependent anomalies is unclear.

Here, we address the two major challenges in the TSAD tasks: **(1) point and contextual anomalies are hard to detect in time series data** and **(2) high-dimensional and noisy time series data are difficult to represent**. We employ deep models to capture complex time series data, and we use self-supervised CL in the TSAD task with multiple point and contextual anomaly-induced temporal transformations. We are different from the classical CL approaches, which usually define the original data instances and their transformations as the *positive pairs*, while data instances with transformations of other instances as *negative pairs*. This approach is practical for learning not task-specific representations; however, it does not target complex

anomaly patterns in time series. Thus, we adapt positive and negative pairs based on the artificial temporal transformations. By artificially augmenting positive and negative temporal transformations, we aim to explicitly encourage the model to learn a robust representation of normal data and to differentiate the normal data from anomalies.

The main contribution of this chapter can be summarized as follows:

1. we propose a CL framework for TSAD
2. we propose multiple temporal time series transformations for the anomaly detection task
3. we demonstrate the effectiveness of the proposed framework with extensive empirical experiments on common real-world TSAD benchmark datasets.

2.2 Related works

2.2.1 Time series anomaly detection

Classical anomaly detection approaches have shown their efficacy in detecting highly deviating points from their neighborhoods in the same data collection, e.g., PCA [PKS18], density-based LOF [BKNS00] and the one-class classifiers [SWSST00, TD04]. Recent deep models expand the anomaly detector to more high-dimensional and complex data. They include extended one-class approach DeepSVDD [RVG+19], reconstruction-based approaching using autoencoders [ZSM+18, ZSZ+21] and generative approaches [SSW+17, AAAB18].

A major challenge in TSAD is detecting contextual anomalies. To capture contextual information, one thread of works is based on time series forecasting and uses the prediction error to indicate anomalies [GRN18, MVS+15]. Several deep models are also used to capture the temporal information in time series data. Malhotra et al.[MRA+16] use LSTMs build Autoencoders to reconstruct time series data. Similarly, Hundman et al. [HCL+18] use LSTMs for forecasting-based anomaly detection. Su et al. [SZN+19] employ GRUs, which are supposed to be easier to train than LSTMs due to their fewer parameters. Convolutional Neural Networks have also been used for time series data. Bai. et al. [BKK18] have shown empirical results that TCNs outperform LSTMs and RNNs in sequence modeling. Thill et al. [TKWB21] construct Autoencoders with TCNs for anomaly detection. Finally, attention mechanism-based Transformer models are capable of long-term sequences. Tuli et al. [TCJ22] propose a Transformer-based anomaly detector with an adversarial training procedure that can amplify reconstruction

errors. Xu et al. [XWWL21] compose a Transformer network with prior- and series-association to learn the temporal dependency.

2.2.2 Contrastive learning

Data augmentation is a key step in CL. In computer vision tasks, geometric transformations enrich the image data without dramatically changing the semantic, e.g., flipping [CSL21], rotation [SLY+21], reflection [GEY18], permutation [TMJS20], jittering [CKNH20], cropping [WBR+20] and changing brightness [WBR+20]. To enrich the transformation beyond the original image into a more general case, Bergman et al. [BH20] employ affine transformation. By manipulating the affine parameters, the affine transformations allow for dimensionality reduction, non-distance preservation, and random transformation. Mistra et al. [MZH16] extend the application to the video domain and transform the data by shuffling the frames.

A common training strategy is the SimCLR[CKNH20], which encourages close embedding of positive pairs and penalizes nearly embedding of negative pairs [TMJS20, WBR+20]. The InfoICE loss [OLV18, dHL21] manipulates the mutual information in the latent space. Incorporated with the downstream classification tasks, the constrictive objective is also combined with cross entropy [SW22], triplet center loss [BH20] and one-class classification loss [SLY+21].

Under the self-supervised setting, CL also shows its strength in various anomaly detection tasks with the biased class distribution. Winkens et al. [WBR+20] enhance out-of-distribution detection with the contrastive objective. Sohn et al. [SLY+21] incorporate negative-sample-free CL with deep one-class classification for anomaly detection. Wang et al. [WLM+22] further propose using distribution augmentation to overcome the class collision problem in the one-class setting.

Despite the recent development of CL, the application domain still needs to be expanded beyond computer vision. Time series data still faces the challenge of getting proper transformations to form contrastive pairs. In the existing works, Shenkar et al. [SW22] transform tabular data by masking consecutive feature subsets; however, they do not consider the temporal dependency. Analog to jittering for image data, Wang et al. [WLM+22] transform time series data by adding noise and manipulating the sequence magnitude. Another implicit solution is to learn transformations with neural networks [QPK+22, SQK+22]. However, they do not explicitly consider the context-relevant anomalies in the time series data during their transformation processes. Considering the unique nature of point and contextual anomalies, we extend the artificial time series transformation (e.g., jittering and pattern-wise magnitude change) by Wang

et al. [WLM+22] with extensive anomaly-induced positive and negative transformations and design a novel self-supervised CL framework for TSAD.

2.3 Methodology

2.3.1 Problem statement

Let $X = \{X_t\}_{t \in \mathbb{Z}}$ be a D-dimensional time series ($X_t \in \mathbb{R}^D$). $W = \{X_{t+1}, ..., X_{t+L}\}$ is a sliding window over X with length L. We aim to detect both point and contextual anomalies from the sliding window. A point anomaly occurs at a single timestamp $T \in [t+1, t+L]$, e.g., a spike. A contextual anomaly event describes a consecutive subsequence of W in time period $[T, T+l] \subset [t+1, t+L]$ that makes W deviate from the common sliding windows. For a given sliding window W, we need to predict one anomaly score a, indicating the likelihood of W being anomalous. As a post-step, a threshold over a can be applied to receive a binary prediction of being anomalous. The selection of such a threshold is out of our scope.

2.3.2 Contrastive anomaly detection

2.3.2.1 Architecture overview

The ContrastAD model consists of two stages. Firstly, we apply positive and negative temporal transformations to each input sliding window W (Section 2.3.2.2). Secondly, after the original and transformed data windows are encoded into the latent space, we conduct the CL process using our contrastive objective (Section 2.3.2.3). The final anomaly score is calculated based on the contrastive loss (Section 2.3.2.4). Figure 2.1 shows an overview of the model architecture.

2.3.2.2 Temporal transformations

We follow the common assumption in unsupervised anomaly detection tasks where pure normal data is available for the training and anomalies only appear in the test phase. To meet the nature of the TSAD problem, we carry out temporal transformations of sliding windows considering the local context within each window. Specifically, we augment the normal time series data windows with the *jittering*-based positive transformation and multiple anomaly-induced negative transformations. The model is generalized by augmenting the normal training data to

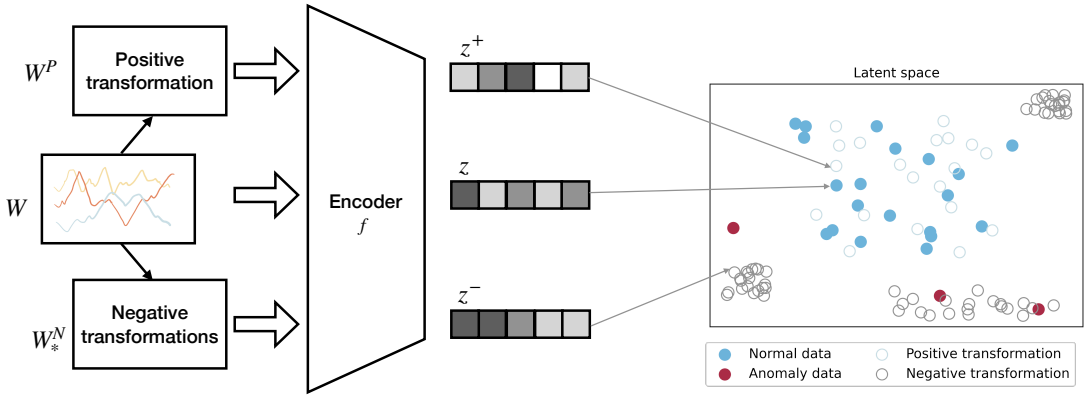


Figure 2.1: ContrastAD overview. For each input window W, we augment it with multiple anomaly-induced positive (W^P) and negative (W^N_*) transformations. After the augmentation, all data are fed to a time series encoder. The embeddings (z,z^+ and z^-'s) are used for the contrastive learning process in the latent space.

learn noisy and slightly perturbed normal patterns. The negative temporal transformations contain both point and contextual anomaly-induced artificial abnormal patterns. The contrastive objective pushes the normal cluster away from negative transformations in the latent space.

In the *jittering*-based positive transformation, for data window $W \in \mathbb{R}^{L \times D}$, we add random noise $\epsilon \in \mathbb{R}^{L \times D}$ to W, specifically,

$$W^P \coloneqq W + \epsilon \tag{2.1}$$

for $\epsilon_{i,j} \in \mathcal{N}(0, \sigma_{jitter})$ with $i \in [1, L]$ and $j \in [1, D]$. σ_{jitter} controls the strength of the noise.

Moreover, oppositely, we augment the data window W with multiple anomaly-induced negative transformations to encourage the normal windows to be embedded apart from deviating patterns. Specifically, we incorporate *spike* to simulate point anomalies and *shuffle*, *trend* as well as *scale* to simulate contextual anomalies. So that we end up with the negative transformation set

$$W^N \coloneqq \{W^N_{spike}, W^N_{shuffle}, W^N_{trend}, W^N_{scale}\} \tag{2.2}$$

The *spike* transformation simulates point anomalies by adding extreme values. The transformation W^N_{spike} is achieved by replacing the value $W_{T,d}$ at random timestamp T and feature d with an extreme value deviating from the mean of the feature d

$$W'_{T,d} = \mu_d + \lambda_{spike} \cdot \sigma_d \tag{2.3}$$

where μ_d and σ_d are the mean and standard deviation on feature d in the sliding window. λ_{spike} is a hyperparameter that controls the spike amplitude.

The *shuffle* transformation simulates an interrupted temporal context by shuffling the first and second half of the window. The data window still contains the local temporal context in each half window; however, the global order within the entire window is shuffled.

$$W^N_{shuffle} = [W_{\lceil \frac{L}{2} \rceil:L}; W_{1:\lceil \frac{L}{2} \rceil - 1}] \tag{2.4}$$

The *trend* transformation simulates abnormal data windows caused by irregular trends within the window. W^N_{trend} scales each timestamp with an incremental factor so that for value $W_{T,d}$ at timestamp T on feature d,

$$W'_{T,d} = (\frac{L+T}{L}) \times W_{T,d} \tag{2.5}$$

where L is the window length.

The *scale* transformation stretches the sequence on its feature dimensions. The transformation scales the whole window W by a random factor $\lambda_{scale} \in \mathcal{N}(2, \sigma_{scale})$:

$$W^N_{scale} = W \times \lambda_{scale} \tag{2.6}$$

where σ_{scale} is a hyperparamter controls the scaling strength.

Figure 2.2 visualizes an example of the temporal transformations on one normal and one abnormal data window from the univariate ECG [CHR+15] dataset.

2.3.2.3 Contrastive objective

All the positive, negative temporal transformations and the original time series windows are fed into the encoder to get latent space representations. The encoder f can be one of the time series representation learning models that capture contextual information. We give a further discussion on the existing encoder models in Section 2.3.2.5. After the encoding process, we get the latent space representations $z = f(W)$ for the original input window W, $z^+ = f(W^P)$ for the positive-transformed window W^P and $z^-_* = f(W^N_*)$ for each negative-transformed window $W^N_* \in W^N$.

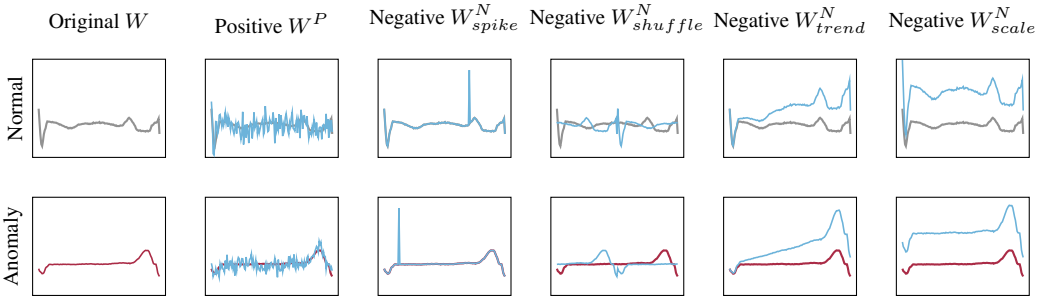


Figure 2.2: An example of the temporal transformations on ECG. The gray lines and the red lines are two examples of the original normal and abnormal data. The blue lines are the temporal transformations.

We define contrastive pairs in each mini-batch $\mathcal{B}$ of data windows. Figure 2.3 visualize an overview of all pairs derived from the two data windows $W(i)$ and $W(j)$ in mini-batch $\mathcal{B}$. Our intuition is to pull the normal windows and their positive transformations together and push positive transformations of different data windows as well as data windows with their negative transformations apart from each other. Concretely, we define positive pairs as the embeddings of a window $W(i) \in \mathcal{B}$ and its own positive transformation. So that the positive loss component is

$$\mathcal{L}^P = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} exp(sim(z(i), z^+(i)))/\tau) \tag{2.7}$$

where τ is the temperature parameter, $sim(\cdot, \cdot)$ denotes the cosine similarity.

Furthermore, in each mini-batch $\mathcal{B}$, we define the negative pairs and corresponding negative loss components as

(1). the embeddings of the positive transformations of two different data windows $W(i)$ and $W(j)$

$$\mathcal{L}_{WW} = \frac{2}{|\mathcal{B}|^2 - |\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=i+1}^{|\mathcal{B}|} exp(sim(z^+(i), z^+(j))/\tau) \tag{2.8}$$

where $\frac{1}{\frac{|\mathcal{B}|(|\mathcal{B}|-1)}{2}} = \frac{2}{|\mathcal{B}|^2-|\mathcal{B}|}$ is the number of distinct negative sample pairs within one mini-batch.

(2). the embeddings of each data window $W(i)$ and its every negative transformation $W^N_*(i) \in W^N$

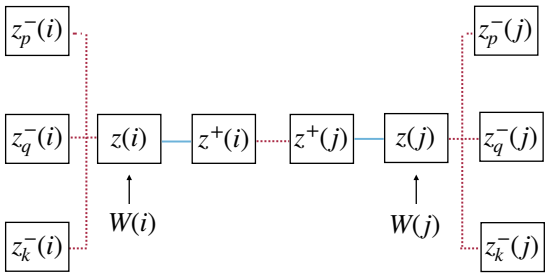


Figure 2.3: Contrastive pairs. $z(i)$ and $z(j)$ are window embeddings of $W(i)$ and $W(j)$. z^+ and z_*^- are corresponding positive and negative transformations. The blue solid lines connect positive pairs, and the red dashed lines connect negative pairs in contrastive learning.

$$\mathcal{L}_{WN} = \frac{1}{|\mathcal{B}| \times |W^N|} \sum_{i=1}^{|\mathcal{B}|} \sum_{p=1}^{|W^N|} exp(sim(z(i), z_p^-(i))/\tau) \tag{2.9}$$

The final contrastive objective is

$$\mathcal{L} = -log \frac{\mathcal{L}^P}{\mathcal{L}^P + \mathcal{L}_{WW} + \mathcal{L}_{WN}} \tag{2.10}$$

2.3.2.4 Anomaly score

We define the anomaly score of each data window based on the contrastive loss [QPK+22] without access to the labels. Specifically, we adapt the positive and negative loss components to extract each data window's positive and negative contributions. Concretely, for data window $W(i)$, the positive contribution is

$$a^P(i) = exp(sim(z(i), z^+(i)))/\tau) \tag{2.11}$$

and the negative contributions are

$$a_{WW}(i) = \frac{1}{|\mathcal{B}| - 1} \sum_{j \in [1,|\mathcal{B}|], j \neq i} exp(sim(z^+(i), z^+(j))/\tau) \tag{2.12}$$

$$a_{WN}(i) = \frac{1}{|W^N|} \sum_{p=1}^{|W^N|} exp(sim(z(i), z_p^-(i))/\tau) \tag{2.13}$$

The final anomaly score is

$$a(i) = -log \frac{a^P(i)}{a^P(i) + a_{WW}(i) + a_{WN}(i)} \tag{2.14}$$

2.3.2.5 Time series encoders

In ContrastAD, the encoder f learns representations of the original data windows and their temporal transformations. Several existing time series representation learning models can act as the encoder. In our experiment, we primarily use the autoregressive model-based Contrastive Predictive Coding (CPC) [OLV18], which is effective in multiple sequential data representation learning tasks [dHL21, Hen20]. This is especially important for high-dimensional and long-span time series to keep the global structure, while classical unimodal approaches are often not powerful enough, and conditional generative models are computationally intense [OLV18]. We compare different time series encoders in Section 2.4.2.2.

2.4 Experiments

2.4.1 Experiment setup

2.4.1.1 Dataset description

We conduct experiments on multiple TSAD benchmark datasets. ECG [CHR+15] is a univariate dataset describing 5000 patient heartbeats in 5 classes. We consider the two smallest classes as anomalies and the other three as normal. SMAP and MSL are high-dimensional spacecraft telemetry data with pre-labeled point and contextual anomalies [HCL+18]. These datasets are noisy and do not contain common repeating patterns. SWaT [MT16] is collected from a water treatment testbed, where artificial attacks are labeled as anomalies. Finally, UCR [KTNA21] is a challenging univariate dataset from various real-world applications. SMAP, MSL, and UCR contain subsets collected from different devices. We train a dedicated model for each subset. In our experiments, we use 5 subsets from UCR ($001 \sim 005$), SMAP ($P-1$, $S-1$, $E-1$, $E-2$, $E-3$) and MSL ($M-6$, $M-1$, $M-2$, $S-2$, $P-10$) respectively based on the order in the original datasets, which cover both point and contextual anomalies. The train and test sets are specified in the original datasets, so the train sets only contain normal data. The evaluation results are averaged over all subsets. The source code of ContrastAD is available online [1].

[1] https://github.com/KDD-OpenSource/ContrastAD

2.4.1.2 Competitors

We compare ContrastAD with multiple common baseline anomaly detection models. We include classical anomaly detection approaches Local Outlier Factor (LOF) [BKNS00], One-Class SVM (OCSVM) [SWSST00] with RBF kernel and Isolation Forest (IF) [LTZ08]. Furthermore, we also compare with the recent reconstruction-based deep models LSTM-Autoencoder (LSTMAE) [MRA+16] and DAGMM [ZSM+18]. The implementation of baseline models is taken either from Scikit-learn [1] or online resources [2].

2.4.1.3 Evaluation metric

Our model calculates the anomaly score based on the contrastive loss. The anomaly score indicates the likelihood that a window is an anomaly. We do not include a specific thresholding technique to receive a binary prediction. The anomaly detector should deliver a good performance of multiple threshold settings, and the user can select the threshold depending on the concrete use cases. Following [QPK+22, GEY18], we use the AUROC (Area Under the Receiver Operating Characteristic curve) to evaluate the anomaly detector performance.

2.4.1.4 Parameter configuration

For sliding window construction, we set window size $L = 140$ for ECG defined by the data source. For the other datasets without prior knowledge, we generally set $L = 100$ except $L = 50$ for the smaller dataset MSL. We slide the window forward without overlap. For the time series transformation, we set hyperparameters by default $\sigma_{jitter} = 0.2$ for W^P, $\lambda_{spike} = 5$ for W^N_{spike} and $\sigma_{scale} = 0.8$ for W^N_{scale}. With these, the spike transformation generates a significant point anomaly, while the *jittering* transformation only augments the data with minor perturbation. Beyond the selected default parameter configuration, an extensive parameter sensitivity analysis is provided in Section 2.4.2.3. For ContrastAD, we train with the Adam optimizer [KB14] for 50 epochs with learning rate 0.001 and batch size $|\mathcal{B}| = 8$. We use 20% of the training data for validation. In the contrastive objective, we set the temperature parameter $\tau = 0.2$. For the time series encoder, in addition to the default CPC model [OLV18], we also compare to a three-layer bidirectional LSTM model, a TCN model [BKK18] with kernel size 5 and a Transformer model [ERC+21], all with 80 hidden units.

[1] https://scikit-learn.org
[2] https://github.com/KDD-OpenSource/DeepADoTS

Table 2.1: Overall performance (AUROC)

	ECG	SMAP	MSL	SWaT	UCR
LOF	0.487	0.348	0.702	0.435	0.502
OCSVM	0.505	0.268	0.789	0.617	0.526
IF	0.500	0.307	0.500	0.469	0.481
LSTMAE	0.566	0.253	0.786	**0.791**	0.535
DAGMM	**0.643**	0.576	0.745	0.659	0.567
ContrastAD (Ours)	0.500	**0.619**	**0.813**	0.729	**0.734**

2.4.2 Performance

2.4.2.1 Overall performance

The overall performance (AUROC score) of ContrastAD and the baseline models has been summarized in Table 2.1. Our model outperforms the baseline model on three benchmark datasets and performs on par with the baseline models on SWaT. Specifically, on the ECG data, the classical deep models LSTMAE and DAGMM show better performance than ContrastAD. This may indicate that the artificial transformations in ContrastAD do not bring many benefits to the dataset containing fixed repeating normal patterns and specific abnormal patterns. In this case, we recommend defining data-specific artificial negative transformations based on prior knowledge of the datasets.

2.4.2.2 Ablation study

We conduct two ablation studies to examine the importance of the negative transformation functions (Table 2.2) and encoder models (Table 2.3). ContrastAD with all four negative transformation functions *spike*, *shuffle*, *trend*, and *scale* shows the best performance on MSL, SWaT and UCR and is on par with the two deep models on SMAP. Especially on UCR, which is claimed to be a challenging dataset [XWWL21], even removing a single transformation function will cause a significant decrease in the AUROC score. The ECG dataset shows results in the opposite direction. Removing *trend* or *scale* does not impact the performance, while removing *spike* or *shuffle* is even beneficial. One possible reason is that ECG contains neither severe point anomalies like *spike* nor contextual anomalies like *shuffle*; those two functions may confuse the model with *unrealistic transformations*. In the other high-dimensional datasets with

Table 2.2: Ablation study: negative temporal transformations (AUROC)

	ECG	SMAP	MSL	SWaT	UCR
w/o spike	**0.535**	0.565	0.705	0.626	0.464
w/o shuffle	0.530	0.474	0.632	0.650	0.455
w/o trend	0.500	0.624	0.710	0.682	0.484
w/o scale	0.500	**0.639**	0.702	0.588	0.553
ContrastAD	0.500	0.619	**0.813**	**0.729**	**0.734**

Table 2.3: Ablation study: time series encoders (AUROC)

	ECG	SMAP	MSL	SWaT	UCR
LSTM	0.430	0.349	0.503	0.365	0.473
TCN	0.449	0.447	0.516	0.670	0.369
Transformer	**0.540**	**0.718**	0.688	0.708	0.477
CPC	0.500	0.619	**0.813**	**0.729**	**0.734**

more general and noisy patterns, the negative transformations benefit the learning procedure.

Table 2.3 shows the model performance when alternating the encoder CPC with another time series representation learning model. The CPC encoder shows dominating performance on MSL, SWaT, and UCR while is on par with other encoders on ECG and SMAP. The classical time series modeling approaches LSTM and TCN, however, do not show convincing results.

2.4.2.3 Parameter sensitivity

We show the results of parameter sensitivity analysis in Figure 2.4. Since the contrastive pairs are built within mini-batches, the batch size is supposed to be an important factor in ContrastAD. We evaluate the model performance under the batch sizes $|\mathcal{B}| \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. We train one model per subset (some datasets do not allow batch sizes larger than 32). The results are shown with mean and standard deviations over three runs. Our experimental results show that large batch sizes do not directly bring better results. Rather, there is a drop in the AUROC score on UCR when the batch size increases from 16 to 128. Most datasets show increasing performance when the batch size increases from 1 to 16. This indicates that a proper middle size of batches helps to learn features among local instances.

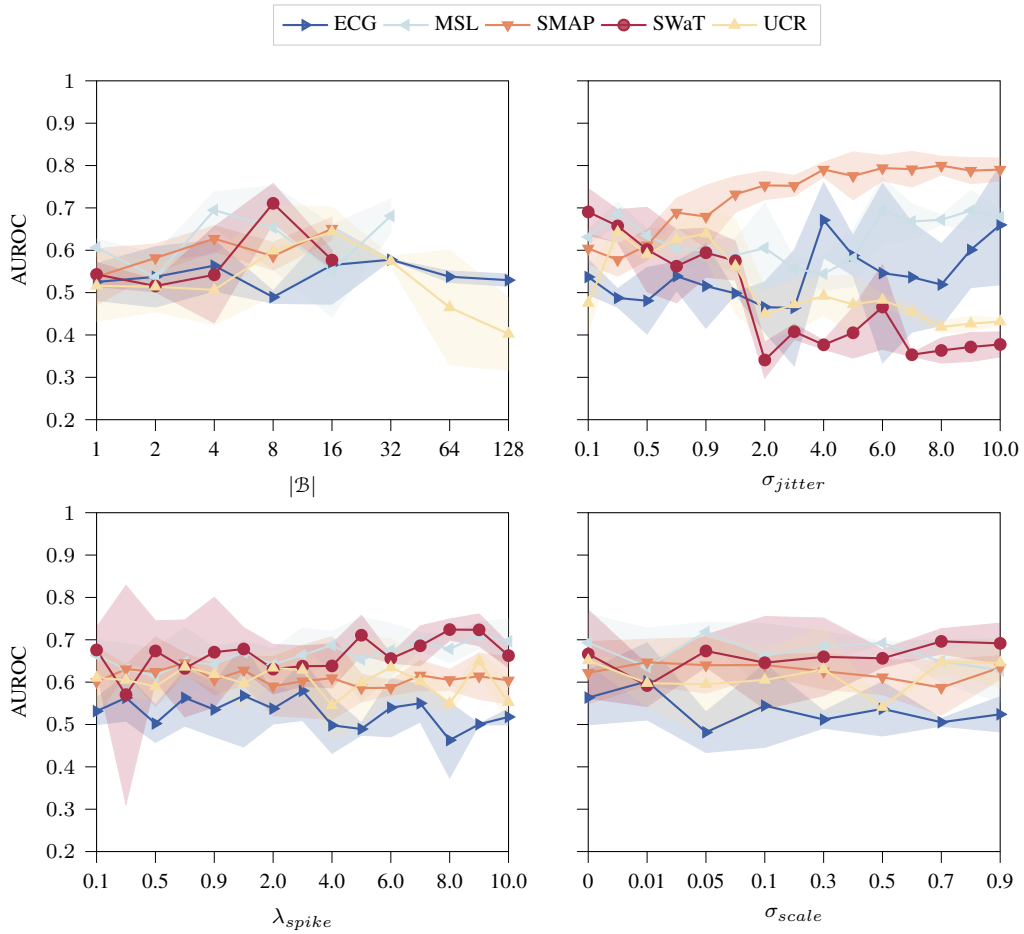


Figure 2.4: Parameter sensitivity analysis. Batch size $|\mathcal{B}|$. Positive transformation parameter σ_{jitter}. Negative transformation parameter λ_{spike}. Negative transformation parameter σ_{scale}.

Furthermore, Figure 2.4 also shows the result of sensitivity analysis of the hyperparameters σ_{jitter}, λ_{spile} and σ_{scale} in the negative temporal transformations. The parameters are used to determine the strength of the transformation effect. We examine both small values $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and large values $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ for σ_{jitter} and λ_{spile}. For σ_{scale}, we try values in $\{0, 0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9\}$. We observe that ContrastAD is generally not sensitive to all parameters in the value ranges we have examined. For the σ_{jitter} in the positive transformation, a small value will add random noise to the input signal, which helps the model to learn robust representations of the normal pattern. However, a very large σ_{jitter} will lead a noisy normal data becoming an anomaly, therefore harming the performance (e.g., on SWaT). On the opposite side, a tiny spike is hard for the model to distinguish. Therefore

we recommend to select values $\sigma_{jitter} \in [0, 1]$ and $\lambda_{spile} \in [1, 10]$. The *scale* transformation scales the data window with some random factors $\lambda_{scale} \in \mathcal{N}(2, \sigma_{scale})$. In our experiments, we fix the average factor value to 2, so it doubly stretches the data window amplitude. The hyperparameter σ_{scale} does not significantly impact performance.

2.5 Conclusion

In this chapter, we presented a novel TSAD framework, ContrastAD, by temporal transformation-based CL. Specifically, we defined multiple point and contextual anomaly-induced temporal transformations when constructing contrastive pairs. Our experimental results indicate that ContrastAD performs better or on par with common baseline models. In the extensive analysis, we discovered that ContrastAD brings more benefits to high-dimensional and noisy datasets without common repeating patterns.

Currently, we include four negative transformations to simulate the most common point and contextual anomalies in time series data. These contribute to the anomaly detector even though the anomalies in the datasets are not directly the same as what we generated. However, we believe few real anomaly data are necessary for the model to learn precise negative transformations for datasets with some common repeating patterns or limited types of anomalies, e.g., ECG. To this end, one promising future work is to develop the temporal transformation procedure in a semi-supervised manner, with a few labeled anomaly patterns as guidance for the negative temporal transformations.

So far, we restrict the problem setting to static time series data, where the training and testing data are uniformly sampled. However, this does not hold in many real-world scenarios. Data property evolves with the environment, and the same goes for anomalies. In such cases, one stationary model is not able to detect anomalies from all temporal contexts. To this end, we continue with data stream anomaly detection in Part III.

Part III

Data Stream Anomaly Detection

3

Correlation Drift Detection

Data streams are time series data collected in an online streaming manner. Therefore, the temporal information between timestamps is also an important concern in data stream anomaly detection. However, different from static time series, dynamic data streams contain distributional drifts along with the environment. Consequently, a single static model as in Chapter 2 may fail. As stated in Section 1.2.2, a major challenge is to detect the distributional drifts and adapt the anomaly detection model according to the latest data characteristics. In Chapter 3, we address the drift detection problem, and in Chapter 4, we continue with the adaptive anomaly detection in data streams.

In the literature, drift detection approaches have addressed this problem in various perspectives [GMCR04, BGdCÁF+06, BG07], however, detecting correlation structure changes of the input dimensions under unsupervised setting is still an open challenge. The different data features before and after a concept drift hinder the performance of most of the predictive models, mostly requiring re-training. Detecting such concept drifts represents a severe problem, as volatile data labeling is often expensive or delayed in streaming data. Moreover, classical concept drift detectors usually struggle with detecting drifts in correlations of multivariate data streams.

In this chapter, we focus on unsupervised drift detection, tracking correlation changes in the input variables without class labels. By introducing the *sliding SHAPley values series* (slidSHAPs), we propose a fully unsupervised drift detector for multivariate data streams with categorical value domains; our approach detects correlation-based drifts through a representation of the correlation structure of the input data. The slidSHAP series also underlines distributional drifts only in a few univariate input variables, thus being more sensitive to drifts than prior drift detection methods. In contrast to the well-known application of Shapley values for interpretable machine learning, we use this foundational game-theoretic concept to extrapolate information on the correlation structure of data streams and achieve higher sensitivity towards multiple drifts in the empirical evaluation of synthetic and real-world data.

3.1 Introduction

Concept drifts are a severe problem in streaming data analysis [LLD+18]. While *supervised drift detection* concerns the drifts w.r.t. the conditional distribution function of a given class label y to the input variables $P(y \mid X)$, *unsupervised drift detection* focuses on the distributional drifts w.r.t. the probability distribution function of the input variables $P(X)$. In this work, we restrict ourselves to unsupervised drift detection; hence, we are unaffected by expensive, incorrect, or delayed information on class labels. In this work, we primarily focus on detecting abrupt drifts.

In many real-world scenarios, drifts are not easy to spot. We look for distributional shifts inducing drifts in the correlation structure of multivariate data streams; generally, those distributional drifts are hard to visualize within the chaotic behavior of the data streams. Through the representation of the time-evolving correlation structures with the slidSHAP series, the drifts are made easy to detect. Moreover, drift detectors fail to analyze complex correlation structures among the data stream input dimensions. State-of-the-art methods limit to covariance evaluation studies [QAWZ15, AK18], and, though a variety of measures [Spe61] try to assess the structure of correlations in sets of N random variables, they often miss the complex multivariate interactions [KMB12, SBS+17]. In most cases, it is insufficient to consider pairwise correlations since more complex multivariate interactions potentially hide within higher-dimensional subsets of variables. In the recent feature importance research, aggregated scores based on Shapley values have been proposed to extract information using information theoretical-based correlation measures [BHMM22], but they are not yet applied in drift detection.

Table 3.1: Categorization and comparison of drift detection methods

		Unsupervised (classifier-free)	Pairwise correlations	Multivariate correlations	Visualization
Prediction error-based	[GMCR04] [BGdCÁF+06] [FBdCÁRJ+14]	✗	✗	✗	✓
Covariance-based	[AK18] [QAWZ15]	(✓)	✓	✗	✗
Shapley value-based	[BSC+21] [ZK20] [ZvdZP+19]	(✓)	✗	✗	✓
slidSHAPs (Ours)		✓	✓	✓	✓

To develop reliable unsupervised drift detection techniques for streaming data, we face two main challenges: **(1) the lack of labels in streaming data** and **(2) the detection of drifts in correlation structure among input variables**. As already mentioned, the latter point can not be reduced to the study of pairwise correlation scores [KMB12, SBS+17, BHMM22]; Moreover, though various methods tackling different kinds of distributional drifts exist [HKR+21, ZvdZP+19, CMO16, AK18, QAWZ15], **none of them detects correlation changes among sets of input variables**. Conversely, many drift detection approaches use the real-time classification error rate with the predicted label as a shift's indicator [GMCR04, BGdCÁF+06, FBdCÁRJ+14], thus requiring labeled data.

We introduce slidSHAPs, an unsupervised method detecting drifts in unlabeled streaming data by tracking the correlation structure of the input dimensions. Our labels-free drift detector focuses on the prior probability distribution function of the input variables: first, we transform the original N-dimensional data stream into a new N-dimensional series representing the correlation structure among the input variables. The drifts in the correlations among the input variables are readily detectable in our slidSHAP series. In contrast to covariance-based approaches, Shapley values aggregate the correlation scores in the subsets of the input variables (see Table 3.1). The slidSHAP series clearly outlines that drifts in a single or a few input variables potentially affect the correlation structure of the whole data stream dimensions. Moreover, one can observe the drifts in the correlation structure on the slidSHAP series; thus, potential drifts in the original streaming data are made directly visible [LBM22]. Finally, we

use the slidSHAP series to detect drifts in streaming data, mapping the concept drifts detected in the series of sliding Shapley values back to the original data. We compare our approach to state-of-the-art unsupervised concept drift detection methods in the experiments, using synthetic and real-world data.

3.2 Related works

Concept drift detectors are often based on the prediction error rate of online classifiers assuming that distributional drifts appear in $P(y|X)$ [HKR+21, BG07, GMCR04]. However, in increasing scenarios, streaming data are collected without labels. The necessity of dealing with distributional drifts on input variables forces the development of unsupervised concept drift detection methods [GCGDS20]. Those methods detect concept drifts by comparing the current data distribution with a reference historical data buffer; they rely on two main steps, i.e., (1) the representation of the latest data stream and (2) the detection of drifts over the representation. Although detection methods can be directly applied to the raw data stream, the complex behavior of streaming data and the massive number of instances can hinder detection performance. Among the several representation approaches, we recall some of the most common ones, e.g., using the mean values of adaptive windows to represent the univariate dimensions [BG07], using linear and non-linear features representing the whole data stream [CMO16] and using multidimensional Fourier transformation to get information from the frequency domain [dCDVdM17]. Recently, combinations of multiple statistical features have been proposed as meta-information vectors [HKR+21]. Other approaches measure the distributional discrepancy between data in different time periods, e.g., the Hellinger distance between two distributions [DP11] and data partition via Kdq-Trees and generalizations of the Kulldorff's spatial scan statistic [DKVY06]. Unfortunately, they do not monitor the correlation drifts among input variables; therefore, feature correlation remains mainly studied using simple covariance. Tracking covariance drifts in a transformed artificial low-dimensional space obtained by applying PCA on the data stream [QAWZ15] has been used to detect concept drifts; however, the approach limits to track covariance drifts in the extracted space. To overcome this limitation, Ahmadi et al. [AK18] use both mean and covariance to represent the concepts in multivariate data streams.

Shapley values [Sha53] are often associated with the interpretability of black-box models; on the contrary, their usage is not limited to trustworthy machine learning. Their popularity

derives from the flexible definition of a *value function* based on which they are computed. Their applications spread in various contexts, from supervised feature selection [CDR07, PHHN16] to the interpretation of black-box models[LL17], from bioinformatics [SMP+20, MFPB10] to the extrapolation of the correlation structure of tabular data [BHMM22]. Their success and popularity in machine learning and bioinformatics are still broadening. As mentioned, Shapley values are not always meant to interpret black-box models but, more generally, to extract information about the role of the players in a game [RWB+22].

As time dependency represents an additional challenge, applying Shapley values in streaming data is still not fairly explored. However, recently, Shapley values started finding applications in streaming data. Guidotti et al. [GMS+20] apply them to explain black-box models on streaming data. TimeSHAP [BSC+21] represents an extension of the Lundberg et al. [LL17] SHAP to time series data; the authors develop a method to compute Shapley values to get event- and feature-level explanations. Antwarg et al. [AMSR21] introduce two different methods based on Shapley values. The first approach detects anomalies through the reconstruction error of an autoencoder, while the second method is based on comparing Shapley values of the original and the reconstructed features. Takeishi [Tak19] studies the difference among Shapley values of single instances before and after a drift in one feature that makes the instance itself anomalous. Nguyen et al. [NLD+19] explain the anomalies by analyzing the gradients to identify the main features affecting the anomalies. Furthermore, some data-specific applications popped out in the literature, e.g., forecasting the income of consulting companies [SMK+21]. Shapley values in data streams are still often trivial transpositions of methods available for prediction tasks circumscribed to either specific contexts or datasets. It is also evident that most attempts are bound to supervised applications, i.e., relying on ground truth labels associated with timestamps. We further recall some works applying Shapley values for drift detection. Among them, Zheng et al. [ZvdZP+19] use Shapley values for drift detection for labeled series, Zhao et al. [ZK20] employ Wasserstein and Energy distances to detect feature drifts without labels; Shapley value and LIME [RSG16] are here used as post-hoc interpretation for the detected drifts. As far as we are concerned, we are pioneering in introducing Shapley values to visualize distributional drifts [LBM22] and, through this work, to detect concept drifts in unlabeled data.

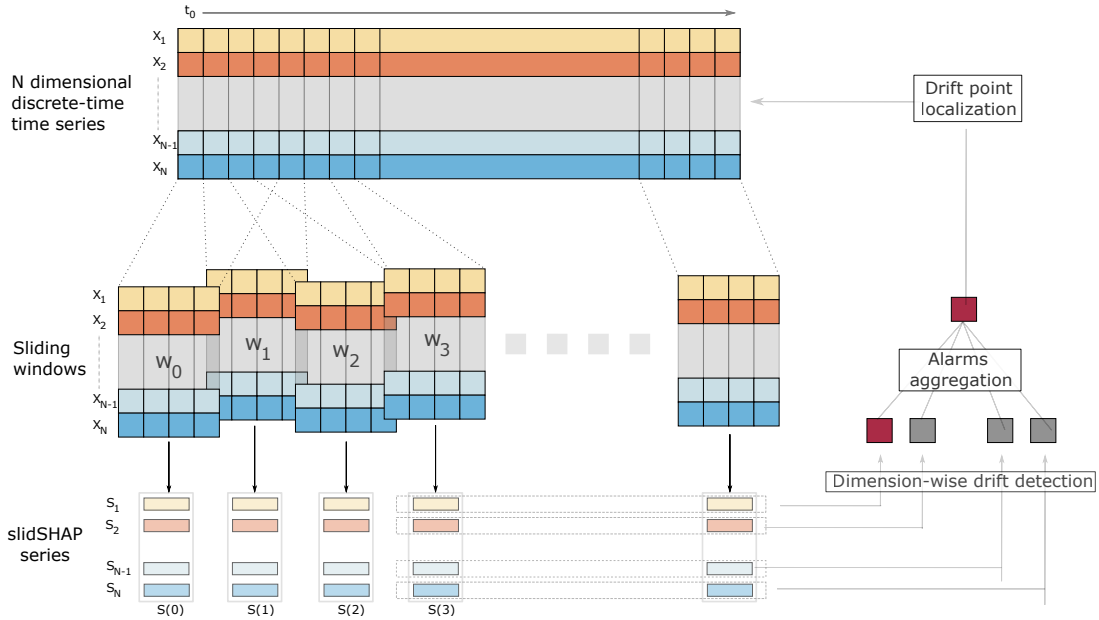


Figure 3.1: slidSHAPs overview: unsupervised concept drift detection in streaming data using slidSHAPs.

3.3 Methodology

We introduce slidSHAPs to visualize and detect correlation drifts in unlabeled multivariate discrete data streams. As we hypothesized, even distributional drifts happening only in a few univariate dimensions can drastically change the correlation structure among all univariate dimensions. We map the data stream into the slidSHAP series, where we implicitly encode correlations among the data stream's input variables as a function of time; the resulting sequence has a different dependency on time from the original timestamps. We use the slidSHAP series to detect concept drifts through statistical tests, and we finally relocate the detected drifts to the original time notion. Figure 3.1 provides an overview of our method; we go through each of the steps in the following sections.

3.3.1 Data stream and sliding windows

We indicate with $X = (X_1, \ldots, X_N)$ a multivariate N-dimensional discrete data stream where X_i is the i-th univariate dimension; we currently restrict the approach to data streams whose dimensions assume only a finite number of values, i.e., either categorical or discrete and finite.

We indicate with t_0 the first timestamp on which the data stream is defined. For each timestamp $t_k > t_0$, $X(t_k)$ is a N-dimensional vector of discrete values, i.e., $X_i(t_k) \in D_i$ and $|D_i|$ is finite. Using this notation, we define *overlapping sliding windows* as a series of time windows $\{w_s\}_{s\in\mathbb{N}}$ dependent on two parameters, i.e., the window length $d \in \mathbb{N}$ and the overlap $a \in \mathbb{N}$ among adjacent windows:

$$w_s = \{t_{s(d-a)}, \dots, t_{s(d-a)+d-1}\}. \tag{3.1}$$

Each window w_s contains d timestamps, and a is the number of timestamps lying in the overlap among adjacent windows, i.e., $|w_s \cap w_{s+1}| = a$ for all $s \in \mathbb{N}$. At the current timestamp t_T we have created $M(T) = \left\lfloor \frac{T-d+1}{d-a} \right\rfloor + 1$ time windows.

Sliding windows are commonly used in concept drift detectors [BG07, BGdCÁF+06, DKVY06]. However, most existing approaches focus on specific statistical features of the sliding windows, which leads to an intrinsic inability to detect drifts in feature correlations. We aim to create a feature extraction function over the sliding windows that outputs representative features with more easily detectable concept drifts. In the following sections, we introduce the slidSHAP series, a novel feature extractor for unlabeled streaming data based on sliding windows.

3.3.2 slidSHAP series creation

Given a multivariate data stream X with N-dimensions, we can interpret the value $X_i(t_k)$ as the realization of a discrete random variable X_i; hence, given the set of timestamps $\{t_0, \dots, t_T\}$, the set $\{X_i(t_0), \dots, X_i(t_T)\}$ contains $T + 1$ independent realizations of the random variable X_i. Similarly, $\{X(t_0), \dots, X(t_T)\}$ is the set of realizations of a N-dimensional discrete random variable. This interpretation allows us to study the correlations among the input variables of the data stream and does not consider the temporal dependency among timestamps.

Given a game, Shapley values represent a way of fairly distributing resources among players [Sha53] and, as already stated, can be used in contexts unrelated to interpretable machine learning. Given a set of players $F = \{X_1, \dots, X_N\}$ and a value function ν, the Shapley values' definition reads

$$\phi(X_i) = \sum_{A \subseteq F \setminus \{X_i\}} k_A \cdot (\nu(A \cup \{X_i\}) - \nu(A)) \tag{3.2}$$

where k_A depends on the number of players N and the size of the set A [Sha53]. Balestra et al. [BHMM22] propose Shapley values within an unsupervised feature selection method.

Given a set of N discrete random variables $F = \{X_1, \ldots, X_N\}$, they propose to interpret F as a set of players and encode the correlations within subsets of features of an unlabelled tabular data set with categorical entries using the Shapley values. The authors claim in favor of using a correlation metric as the value function. They propose for their categorical context to use the total correlation as the value function, i.e., $\nu(A) = H(A) - \sum_{X \in A} H(X)$ where $H(\cdot)$ is the discrete Shannon entropy. The proposed encoding enables information to be extracted from the data set based on the correlation structure. Features obtaining high Shapley values are highly correlated with subsets of other variables, while features with lower Shapley values are uncorrelated with the resting variables. The authors use the Shapley values to rank the features with respect to their ability to represent the correlation structure of the entire data set.

We interpret the realizations of a data stream on a time window $\{t_k, \ldots, t_{k+d-1}\}$ as a discrete tabular data set with N columns and d rows. This allows us to compute a Shapley value for each column, i.e., for each univariate dimension of the data stream using [BHMM22]. Our goal is to detect concept drifts appearing in the input dimensions of the data stream. We trace the concept drifts using the Shapley values of the data stream's input variables when restricted to the sliding windows $\{w_s\}_{s \in \mathbb{N}}$ from Equation (3.1). For each window w_s, we deal with d observations of the N-dimensional random variable X, thus working with a discrete (categorical) tabular data set with N columns and d rows. For each dimension X_i, we get the Shapley value $S_i(s) = \phi(X^i_{w_s})$, i.e., the Shapley values of the input variable X_i when we restrict the observations to the time window w_s. $\phi(X^i_{w_s})$ considers the correlations of X_i with the other dimensions $X^j_{w_s}$ of the data stream in w_s.

Definition 1 (slidSHAP value)**.** For each w_s, we obtain a N-dimensional real-valued vector

$$S(s) = [S_1(s), \ldots, S_N(s)] \tag{3.3}$$

and refer to it as the *slidSHAP value*.

Definition 2 (slidSHAP series)**.** We define the sequence $\{S(s)\}_{s \in \mathbb{N}}$ as the *slidSHAP series*.

In Section 3.3.1, we have introduced the time-dependent sliding windows $\{w_s\}_{s \in \mathbb{N}}$; thus, the slidSHAP series inherits the time-dependency from the windows and not the same time notion as the original data stream. Figure 3.1 represents a visual schema for the slidSHAP series construction process; Algorithm 1 shows the pseudo-code. We extrapolate information about the input dimensions' correlation structure from the original discrete data stream X with discrete finite values and transfer the drift detection problem to a new N-dimensional real-valued

Algorithm 1 Pseudo-code for the slidSHAP series computation.

Input: N-dimensional data stream X, sliding window length d, overlap among adjacent sliding windows a, current timestamp T

$s \leftarrow 0$
$S \leftarrow [\,]$
while $s \leq \left\lfloor \frac{T-d+1}{d-a} \right\rfloor + 1$ **do** ▷ sliding on windows
  for $i \in \{1, \ldots, N\}$ **do** ▷ iterate over dimensions of X
    $w_s = \{t_{s(d-a)}, \ldots, t_{s(d-a)+d-1}\}$
    $S_i(s) = \phi(X^i_{w_s})$
    $S \leftarrow S_i(s)$
  end for
  $s \leftarrow s + 1$
end while
return S ▷ slidSHAP series

series. We interpret the slidSHAP series as a projection of the time-dependent correlation structure of the original data stream. Note that the sliding windows are partly overlapping: given two indices i, j, the intersection $w_i \cap w_j$ is non-empty if they are sufficiently close; hence, the information covered by $S(i)$ and $S(j)$ relates to X on partly overlapping time windows. The setup of the parameters a and d is essential to modulate the *granularity* of the slidSHAP series.

3.3.3 Concept drift detection

The slidSHAP values are based on the distributions and the correlations among the univariate dimensions of the data stream and are label-independent. Oftentimes, when dealing with real-world data streams, only a few input variables are subject to distributional drifts. However, those distributional drifts could affect the correlation structure of all the input variables. Moreover, the concept drifts are often hardly visually detectable, especially when they do not directly affect specific statistical features in which the variables vary.

We expect that distributional drifts in the input variables modify the correlation structure of X on the sliding windows, and the slidSHAP values encode the correlations among the univariate dimensions of X through time. Eventually, the slidSHAP series reflects these distributional drifts allowing us to use it to detect drifts in X.

We target drift detection using statistical tests under the i.i.d. assumption of the slidSHAP values. Similar to the original data stream X, the slidSHAP series is unlabeled; therefore, we

have access only to its dimension-wise distributions to detect drifts. We employ two statistical tests to examine for distributional drifts on each dimension S_i

- the **Student's t-test**
- and the **Kolmogorov-Smirnov test** (or KS-test) [Con99, dRFMB16].

We analyze their performances in the empirical evaluation. Both tests check whether there is statistical significance for two slidSHAPs samples being drawn from the same distribution. We consider a reference sequence of slidSHAP values S_{ref} of length m ending at $s \in \{0, \ldots, M(T)\}$ and a latest new sequence S_{new} with length n precedent to it, i.e.,

$$\mathcal{S}_{\text{new}} = \{S(s) \mid s \in \{s - n + 1, \ldots, s\}\} \tag{3.4}$$

$$\mathcal{S}_{\text{ref}} = \{S(s) \mid s \in \{s - m - n, \ldots, s - n + 1\}\}; \tag{3.5}$$

We call the two data sequences of slidSHAP values *buffers*. F_{ref} and F_{new} are respectively the empirical cumulative distribution functions of $\mathcal{S}_{\text{ref}}$ and $\mathcal{S}_{\text{new}}$ and we test whether there is statistical significance of $\mathcal{S}_{\text{new}}$ and $\mathcal{S}_{\text{ref}}$ to be drawn from the same probability distribution. The user can define the sizes of the new and reference buffers.

We deal with multiple *testing corrections* as we perform a number N of statistical tests (one for each univariate dimension of the slidSHAP series). Among the various multiple hypothesis corrections available, we stick to the Bonferroni correction [BA95, BH95, RGL19], i.e., the null hypothesis is rejected if the minimum p-value among all N tests is smaller than $\frac{\alpha}{N}$. For each $s \geq \min\{m, n\}$, we conduct a set of such dimension-wise statistical tests and compare the performances using the two statistical tests in Section 3.4.2.4. For drift detection, statistical tests are commonly applied on the original data stream [DKVY06, dRFMB16, YWP18]; the inventive step we have introduced is detecting drifts in the slidSHAP series and then relocating them to the original timestamps of the streaming data. The following section presents how to transfer the detected concept drifts to the original timestamps.

3.3.4 concept drifts aggregation and re-location

Due to the construction of the sliding windows, each abrupt concept drift in X is covered by multiple sliding windows. Hence, we need to aggregate the detected drift events on the slidSHAP series to rebuild the single drift event on X. The concept drift detection over the slidSHAP series results in alarms on the corresponding sliding windows and not on single

timestamps. Our ultimate goal is to relocate the drifting positions to the timestamps where they initially take place. We introduce the concept drift *aggregation* and *re-location* in this section.

The sliding windows have length d and overlap a. Given a concept drift at t_{drift} in X, the number of windows containing information about t_{drift} is $\lfloor \frac{d}{d-a} \rfloor > 1$ and the corresponding concept drifts in the slidSHAP series are going to be tested in $\lfloor \frac{d}{d-a} \rfloor + m + n$ statistical tests by moving $\mathcal{S}_{\text{ref}}$ and $\mathcal{S}_{\text{new}}$ one step a time forward on each univariate dimension. Being aware that it is less likely to detect statistical significance for the presence of distributional drift when testing the first and last slidSHAP values involved in the drift event, we first detect alarms on each univariate dimension of the slidSHAP series then we aggregate the alarms using testing correction. At this point, for each window w_s, we have detected an aggregated p-value; due to the dilating effect of drift events in slidSHAP series, we check for sequences of p-values being below the significance level α, i.e., we trigger the concept drift alarm if and only if we find a continuous sequence of $(m+n) \cdot \gamma$ corrected p-values below α. The parameter $\gamma \in \mathbb{R}_+$ scales the number of sequential corrected p-values to be below α before producing an alarm on X and typically ranges in $\left[\frac{1}{2}, 1\right]$. When $(m+n) \cdot \gamma$ rejections of the null hypothesis are detected in sequence, we get an alarm at the current time window w_T. Finally, we relocate the last timestamp $\tilde{t}_{\text{drift}}$ of w_T as the corresponding concept drift position in the original streaming data X.

We underline that we only consider abrupt drifts in the data stream, i.e., the distributional drifts happen in specific timestamps that need to be located. The dilating effect makes abrupt drift in X incremental drifts in the slidSHAP series. The same holds for gradual and incremental drifts in the original data stream, such that our model is easily extendable to non-abrupt concept drifts.

3.3.5 Complexity analysis

The slidSHAP series computation inherits the exponential runtime from the Shapley values computation. The computation of the slidSHAP series has a complexity of $\mathcal{O}\left(d \cdot 2^N \cdot \frac{T-d}{d-a}\right)$ where T is the number of instances to process, N is the number of input variables, d and a are the length and overlap of the windows. The complexity $\mathcal{O}\left(d \cdot 2^N\right)$ derives from the Shapley values' computation [BHMM22]. However, several approximation techniques can be applied [CGT09, MCFH22, vCHHL18, BC21, CKL22] thus accelerating the computation of

the entire slidSHAP series to polynomial time $\mathcal{O}\left(d \cdot N \cdot \frac{T-d}{d-a}\right)$. On the other hand, dimensionality reduction approaches [HKR+21, QAWZ15] can also contribute to reducing d to a manageable scale.

3.4 Experiments

We evaluate slidSHAPs on both concept drift detection and visualization of correlation changes. We compare our model against a set of selected representative unsupervised concept drift detection methods. We consider several synthetic and real-world time series datasets with discrete values. We use ground truth labeling for the allocation concept drifts in the data streams. We acquire a binary set of concept drift alarms after fixing the significance level α, and we compare them with the real timestamps of concept drifts. In summary, we evaluate (1) the concept drift detection performance and alarm delay in Section 3.4.2.1, and (2) the visualization of the drift events in the slidSHAP series in Section 3.4.2.2. The source code of slidSHAPs is available online [1].

3.4.1 Experiment setup

3.4.1.1 Dataset description

We empirically evaluate slidSHAPs with both synthetical and real-world datasets. Table 3.2 summarizes their main characteristics. In the synthetic datasets, we include correlation drifts in the input variables to evaluate the sensitivity of slidSHAPs in detecting different kinds of correlation drifts. We artificially build distributions for each univariate random variable, including correlations among them. Then, we concatenated different subsets at specific timestamps t_{drift}; the input variables follow a correlation structure till t_{drift} and another from $t_{\text{drift}} + 1$ for each concept drift. We constructed two types of synthetic datasets: Type I includes datasets with only 5 features, where the correlations are of a specific type; Type II includes one dataset with random types of correlation drifts at the drift positions and contains 10 features. Each synthetic dataset is constructed as follows: we generate 6 different distributions, each containing 5000 instances, and concatenate them to simulate 5 concept drifts.

Type I synthetic datasets. For each distribution (i.e., concept), we select 2 or 3 variables to be involved in the drift event, while at least one variable always follows the same distribution.

[1] https://github.com/KDD-OpenSource/slidshaps

Table 3.2: Dataset description. Type I datasets are {ADD, MUL, COE, AND, OR, XOR}. Type II dataset refers to MIX.

	Synthetic			Real-world			
	Type I	Type II	LED	BC	PH	KDD	MSL
Instances	30000	30000	90000	286	33659	16000	9809
Variables	5	10	9	10	7	40	54
Categories	$1 \sim 10$	$1 \sim 10$	2	$2 \sim 11$	$4 \sim 13$	$1 \sim 7$	$1 \sim 2$
Drifts	5	5	9	3	5	7	4

Each random sampled variable varies in the set of integers $\{1, \ldots, 10\}$. In ADD, initially $X_3 = X_1 + X_2$, while X_1, X_2, X_4 are independently randomly sampled; after each concept drift, the relation among X_1, X_2, X_3 changes, for example, to $X_2 = X_1 + X_3$. MUL contains only multiplicative relation among X_1, X_2 and X_3; and drifts in a multiplication relations, for example, $X_3 = X_1 \cdot X_2$. In COE, we included linear combinations, for example, $X_3 = c_1 X_1 + c_2 X_2$ where c_1 and c_2 assume various values. Furthermore, we create some datasets, including logical feature correlations. In AND, OR, and XOR, one binary variable depends on the value of the other two variables; after fixing a value c, we include correlations of the type

- $X_3 = \mathbb{1}[(X_1 > c) \& (X_2 > c)]$ in AND,
- $X_3 = \mathbb{1}[(X_1 > c) \mid (X_2 > c)]$ in OR,
- $X_3 = \mathbb{1}[(X_1 > c) \neq (X_2 > c)]$ in XOR

where $\mathbb{1}$ is the boolean function that returns 1 in the case the condition is satisfied and returns 0 otherwise.

Type II synthetic dataset. The dataset MIX contains 10 input variables where X_1, X_2, X_3 are randomly sampled from $\{1, \ldots, 10\}$. For each distribution, all the other variables are constructed using one randomly chosen correlation function of X_1, X_2, X_3 among *addition*, *multiplication*, *linear combination*, *and*, *or* and *xor*. Thus MIX contains a mixture of the correlation drifts. In addition to these synthetic datasets, we also consider one commonly used synthetic dataset in literature, containing drifts in the data distribution instead of explicitly in feature correlation; the LED dataset [FBdCÁRJ+14, PVP18] describes the digit displayed on a seven-segment LED display. A binary 7-dimensional binary vector represents a digit; it contains in

total 9 concept drifts, one per every 10000 instances. Each subset contains the vectorial representations of the ten single digits except one; the drift events are implemented by changing the absent digit.

Real-world datasets. Finally, we included some real-world datasets, processing them as data streams and including concept drifts by concatenating different subsets [Ho05]. We use the following publicly available categorical datasets:

- *Breast Cancer dataset* BC [DG17]
- *Poker Hand dataset* PH [DG17]
- *KDD Cup 99 dataset* KDD [DG17]
- *Mars Science Laboratory dataset* MSL [HCL+18]

BC contains purely categorical features describing breast tumors of patients. We concatenate subsets of patients in different age groups to simulate concept drifts. PH contains one million randomly drawn poker hands. Five features describe the 4 possible suits, and another five features describe the 13 possible ranks. We create virtual drift as in [BPRH13] by sorting the ranks and suits and take a subset with 33659 instances for our experiments. KDD contains both numerical and categorical features describing instances of network intrusion records. We use all features in our experiments and discretize the numerical features into five categories. A subset with 16000 instances from HTTP and SMTP services is selected and the concept drifts are created by concatenating instances from the two services. Finally, MSL is introduced in Section 2.4.1.1. We discard the numerical telemetry values and only consider the 54 remaining binary features. The data is collected from different channels, and we consider the channel changes as concept drifts.

3.4.1.2 Competitors

We compare the performances with some well-known unsupervised concept drift detectors. Here, we give a brief overview of these methods.

Various univariate unsupervised drift detectors exist based on statistical features or distribution discrepancies. Among them, PCA-CD [QAWZ15] detects drifts by computing the divergence metrics on the data's principal components. First, the principal components are computed on a reference window, and samples from a new window are projected onto them.

The result of the divergence metric between the scores for the reference and test window is used as a discriminator factor: if a fixed threshold is reached, a concept drift is alarmed. Based on the Kullback-Leibler divergence, the Kdq-Tree concept drift detection method [DKVY06] partitions data by constructing a Kdq-Tree. The output score of the comparison between the reference window and the test window is also compared to a threshold. Finally, ADWIN [BG07] uses adaptive sliding windows to detect drifts by keeping updated statistics. The discriminator factor here is the difference among the averages of the collected statistics over the reference and new buffers; the obtained score is compared against a threshold. However, all these univariate drift detector methods often fail to detect correlation drifts without significant deviations in the specific statistical features they track (e.g., mean and variance, among others). We only report the results of ADWIN to represent their similar performance. Additionally, HDDDM [DP11] is a batch-based approach that compares the Hellinger distance between the reference and the current batch of data. It considers multi-dimensions as a whole and can detect both abrupt and gradual drifts.

3.4.1.3 Evaluation metric

We evaluate the performance of slidSHAPs and the competitors for unsupervised concept drift detection. The actual timestamps of concept drift t_{drift} are known in each dataset for evaluation, while t^*_{drift} represents the detected drift position. Following Pesaranghader et al. [PV16, PVP18], we introduce an *acceptable delay length* Δ to determine the true positive TP, false positive FP, and false negative FN of detected concept drifts. Whether t^*_{drift} is a TP, FN, or FP is determined by the relative position of the labeled concept drift t_{drift} and the detected position t^*_{drift}. A concept drift alarm is a TP if it belongs to the interval set $\{t_{\text{drift}}, \ldots, t_{\text{drift}} + \Delta\}$, i.e., the delay characterizing its detection is smaller or equal to the accepted delay Δ.

For the evaluation, we use drift detection performance metrics, such as precision, recall, and F1-score, and the average delay, defined as

$$\text{avgDELAY} = \sum_{\text{concept drift} \in \text{ TP}} \frac{t^*_{\text{drift}} - t_{\text{drift}}}{\text{number of TP}}. \tag{3.6}$$

3.4.1.4 Parameter configuration

For the statistical tests, we set the reference and the new buffer sizes $m = n = 10$ and $\gamma = 1.0$, such that the statistical tests are based on sufficient data instances while keeping the prediction

delay low. We conduct the test at a significance level $\alpha = 0.01$. The *acceptable delay length* Δ is set to $5 \cdot d$ where d is the sliding window length, namely all concept drift alarms within 5 window size after the real concept drift are considered as true positives. In all experiments, we implemented the t-test and the KS-test; by default, we opted to report the results obtained using the t-test. Section 3.4.2.4 provides a comparison between the two tests.

In slidSHAPs, the two parameters, window length d and overlap a, influence the construction of the sliding windows and are set through an empirical evaluation of the data stream. We conducted experiments using $\{10\%, 20\%, ..., 90\%\}$ as nine different overlap rates $\frac{a}{d}$ with fixed windows length d. The empirical evidence suggests keeping the overlap rate in the range of $50\% - 80\%$. Further details can be found in Section 3.4.2.3. In the experiments, we construct the slidSHAP series using a fixed window length $d = 100$ and overlap $a = 70$; for the small dataset BC, we set $d = 10$ and overlap $a = 8$. For the real-world high-dimensional datasets KDD and MSL, we compute the slidSHAP series using the upper-bound approximation in [BHMM22].

For HDDDM, we set the data batch size to be two time windows for each dataset. Looking at the performances, we notice that HDDDM shows a generally large delay in detecting drifts; therefore, for HDDDM, we implement a relaxation of the criterion, such that true positives are detected using $\Delta = 10 \cdot d$. ADWIN works by detecting drifts on univariate data streams. Therefore, we train one model for each input data dimension and let them run in parallel for the N dimensions of the streaming data. An alarm is triggered if one concept drift is detected on at least one dimension. For the other competitors Kdq-Tree and PCA-CD, we use the default parameter setting from the GitHub implementation [1]. All the experiments have been run on Intel Xeon CPU E5-2640 v4 @ 2.40GHz with 10 cores.

3.4.2 Performance

3.4.2.1 Overall performance

Table 3.3 shows the overall performance comparison. The slidSHAPs outperforms the competitors with respect to the F1 score in all datasets except MSL; slidSHAPs also shows dominating performance on average delay in most synthetic datasets. Moreover, slidSHAPs appears more sensitive in detecting different types of correlational drifts than the other distribution-based detectors. Since there is no drift in the mean value, ADWIN fails to find any concept drift in each

[1] https://github.com/mitre/menelaus

Table 3.3: Performance summary: for each dataset, the largest F1 score is underlined; the minimum detection delay in each dataset is in bold. In the gray-shaded area, ADWIN can not detect any concept drift.

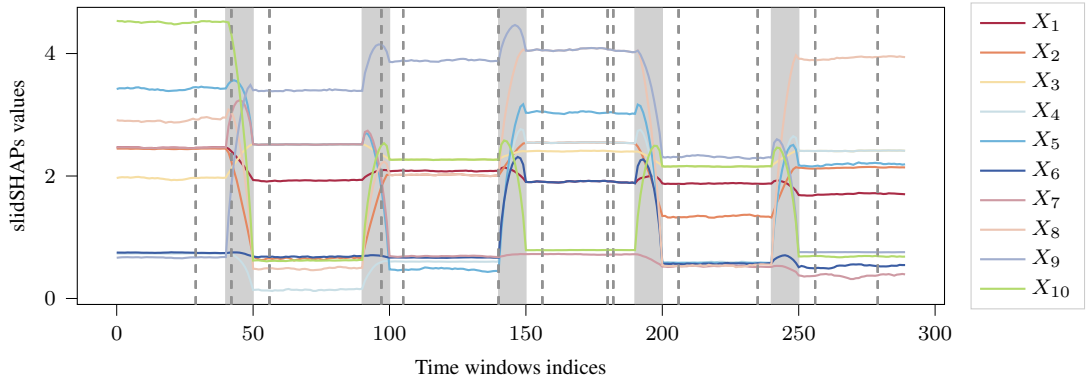
		slidSHAPs (Ours)				HDDDM				ADWIN			
		P	R	F1	avgDELAY	P	R	F1	avgDELAY	P	R	F1	avgDELAY
synthetic	ADD	0.800	0.800	0.800	**384 ± 55**	0.200	0.200	0.200	599 ± 0	-	0.000	-	-
	MUL	0.571	0.800	0.667	**369 ± 70**	-	0.000	-	-	-	0.000	-	-
	COE	0.800	0.800	0.800	**399 ± 40**	0.111	0.200	0.143	559 ± 0	-	0.000	-	-
	AND	0.667	0.400	0.500	**359 ± 0**	0.222	0.400	0.286	599 ± 0	-	0.000	-	-
	OR	1.000	0.400	0.571	**299 ± 0**	0.167	0.200	0.182	559 ± 0	-	0.000	-	-
	XOR	1.000	0.400	0.571	**329 ± 0**	0.250	0.200	0.222	399 ± 0	-	0.000	-	-
	MIX	0.714	1.000	0.833	399 ± 91	0.143	0.400	0.211	**199 ± 0**	-	0.000	-	-
	LED	0.333	0.667	0.444	424 ± 47	0.063	0.556	0.114	199 ± 0	0.111	0.111	0.111	**167 ± 0**
realworld	BC	0.750	1.000	0.857	37 ± 2	1.000	0.667	0.800	34 ± 18	1.000	0.667	0.800	**16 ± 16**
	PH	1.000	0.400	0.571	**382 ± 10**	0.133	0.400	0.200	427 ± 235	-	0.000	-	-
	KDD	0.800	0.571	0.667	455 ± 50	0.350	1.000	0.519	143 ± 90	0.064	1.000	0.121	**36 ± 11**
	MSL	0.500	0.250	0.333	487 ± 0	0.333	0.250	0.286	**117 ± 0**	0.667	0.500	0.571	237± 96

synthetic dataset except LED, i.e., recall equals 0. On the other hand, ADWIN outperforms the slidSHAPs on the MSL dataset, which inherently contains correlation and distributional drifts w.r.t. other statistical features. HDDDM only shows comparable results on BC and performs significantly worse on other datasets.

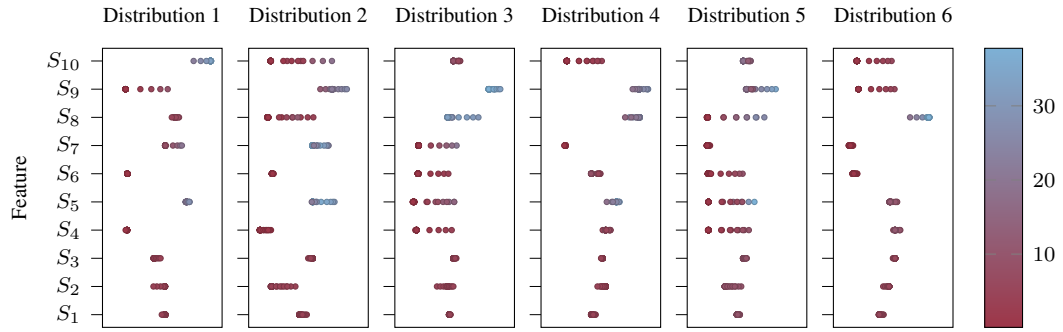
Regarding the average delay of the various methods, ADWIN predicts every incoming instance, generally showing a low average delay. Instead, the slidSHAPs detect the drift on every incoming slidSHAP value, which intrinsically has a delay given by the sliding windows of length d; this mechanism causes our approach to detect concept drifts with a larger delay. HDDDM waits for every batch of data, and consequently, it shows the largest average delay.

3.4.2.2 slidSHAP series analysis and visualization

After fixing the length and the overlap among the sliding windows, the slidSHAP series represents the correlations among univariate dimensions of the data stream. The univariate dimensions of the slidSHAP series follow more distinguishable trends than the original data stream. Although we do not claim that slidSHAPs is an interpretable feature extraction approach, the slidSHAP series provides intuitive visual information of where the concept drifts could be located before statistically checking for their existence.



(a) The vertical dashed lines are the detected concept drifts, and the gray-shaded areas are the changing areas in which the concept drifts in the MIX dataset are mapped.



(b) Evolution of the slidSHAP values ($d = 1000$, $a = 900$) in the different concepts. The color bar indicates the average values of the corresponding dimension in the data stream while the x axis represents the distribution of the slidSHAP values.

Figure 3.2: slidSHAPs visualization for the MIX dataset.

In Figure 3.2, we explore the visual properties of slidSHAPs series using the MIX dataset ($d = 1000$ and $a = 900$). In Figure 3.2a, solid lines are the univariate dimensions of the slidSHAP series, and the gray-shaded areas are the windows in which the concept drifts are mapped using the slidSHAP. The MIX dataset only contains abrupt distributional drifts. However, the slidSHAP series shows smooth changes between one distribution and the next: abrupt drifts are expanded in the slidSHAP series to multiple subsequent time windows. Furthermore, as the slidSHAP values are an aggregation of the correlation structure in subsets of the data stream dimensions, drifts in the slidSHAP series dimensions are observed for all the dimensions and also the ones not affected by significant modification drifts (e.g., X_1, X_2 and X_3). Following the style of [LEC+19], Figure 3.2b represents how the slidSHAP values change in the different distributions. We plot the slidSHAP values against the average value assumed by

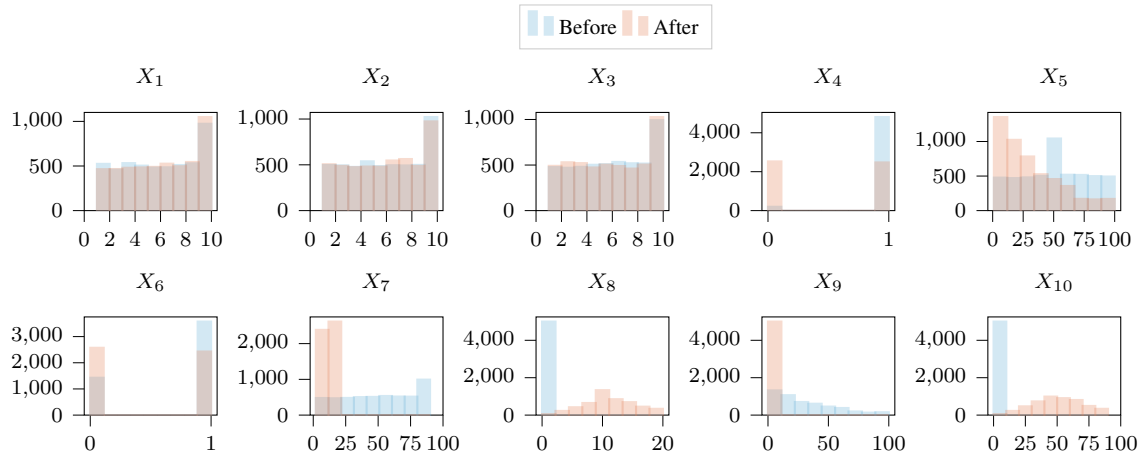


Figure 3.3: MIX dataset analysis: distributions before and after the first concept drift.

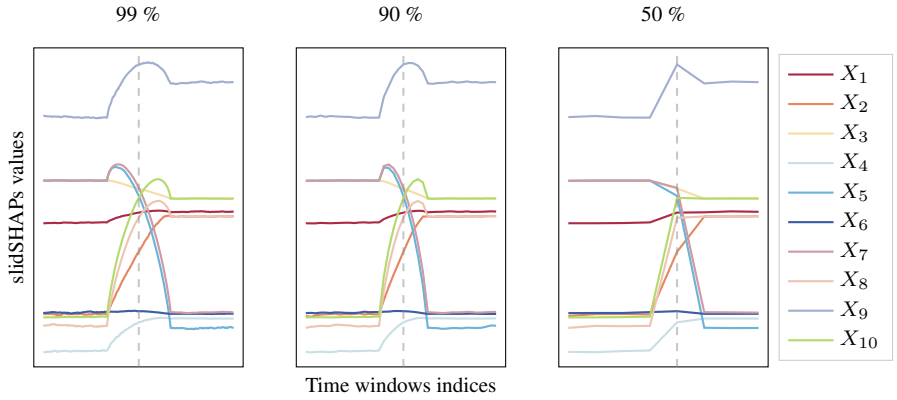


Figure 3.4: MIX dataset analysis: visualization of first concept drift. Solid lines are the univariate dimensions of the slidSHAP series ($d = 1000$ and $99\%, 90\%, 50\%$ overlap rate); the dashed gray line depicts the concept drift position.

the original data stream univariate dimensions. Intuitively, a distributional drift in the input space causes a change in the slidSHAPs value, which can be detected as a distributional drift. The slidSHAP series also shows some unobservable input space concept drifts, where the amplitude of features stays in the same range while the feature correlation changes. In such cases, significant drifts can still be observed in the slidSHAPs values. The drifts in the data stream are highlighted in the slidSHAP series, and the evolution of its univariate dimensions can be used to simplify the detection of such concept drifts in the streaming data.

In Figure 3.3, we plot the different distributions of the streaming data before and after the

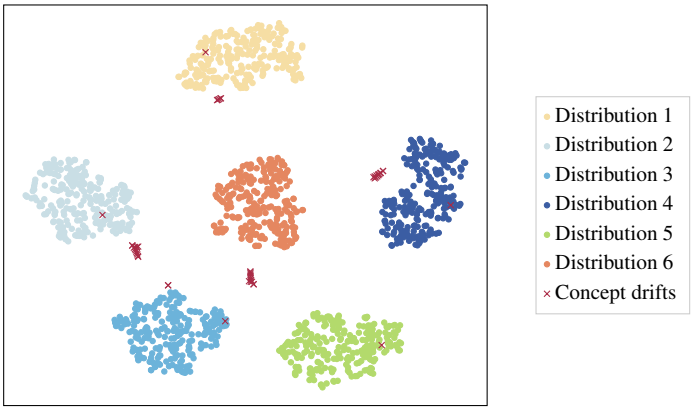


Figure 3.5: MIX dataset analysis: tSNE plot slidSHAP series. Each colored dot represents a concept. The red crosses are the slidSHAPs of the drift events.

first concept drift for each univariate dimension of the MIX data stream. Figure 3.4 shows the evolution of the slidSHAP values around the first concept drift in MIX. We keep using $d = 1000$ and vary the overlap rate $\frac{a}{d}$ among the sliding windows in $\{99\%, 90\%, 50\%\}$. Although the overall behavior of the sliSHAPs is similar using the various parameter settings, a difference in the smoothness in the slidSHAP series in the changing area is immediately noticed. The setting of the parameters d and a also influences the computation time of the whole slidSHAP series (as described in Section 3.3.5). It is worth noting that, to compute the Shapley values using the total correlation, the number of instances per time window should not be too low (e.g., under 100 instances). On the other hand, as the KS-test checks for samples being drawn from equal distributions, if the changes in the slidSHAPs are too smooth, i.e., highly overlapping sliding windows, the test loses statistical power, provoking a higher number of false positives. We can maximize the KS-test's statistical power to get the highest accuracy by selecting a balanced ratio among a and d (c.f. Section 3.4.2.3). Finally, Figure 3.5 visualizes the 10-dimensional slidSHAP values of the MIX data in a two-dimensional space using tSNE. As depicted by the color-coded dots, data from different distributions are well-separable using their slidSHAP representations. The slidSHAP values, whose corresponding sliding window overlaps the drift position (red crosses), lie mostly apart from any cluster. Some are not well-distinguishable with clusters, as they correspond to the beginning and ending sliding windows of the drift events and, therefore, do not show a significant difference to the previous or upcoming distribution.

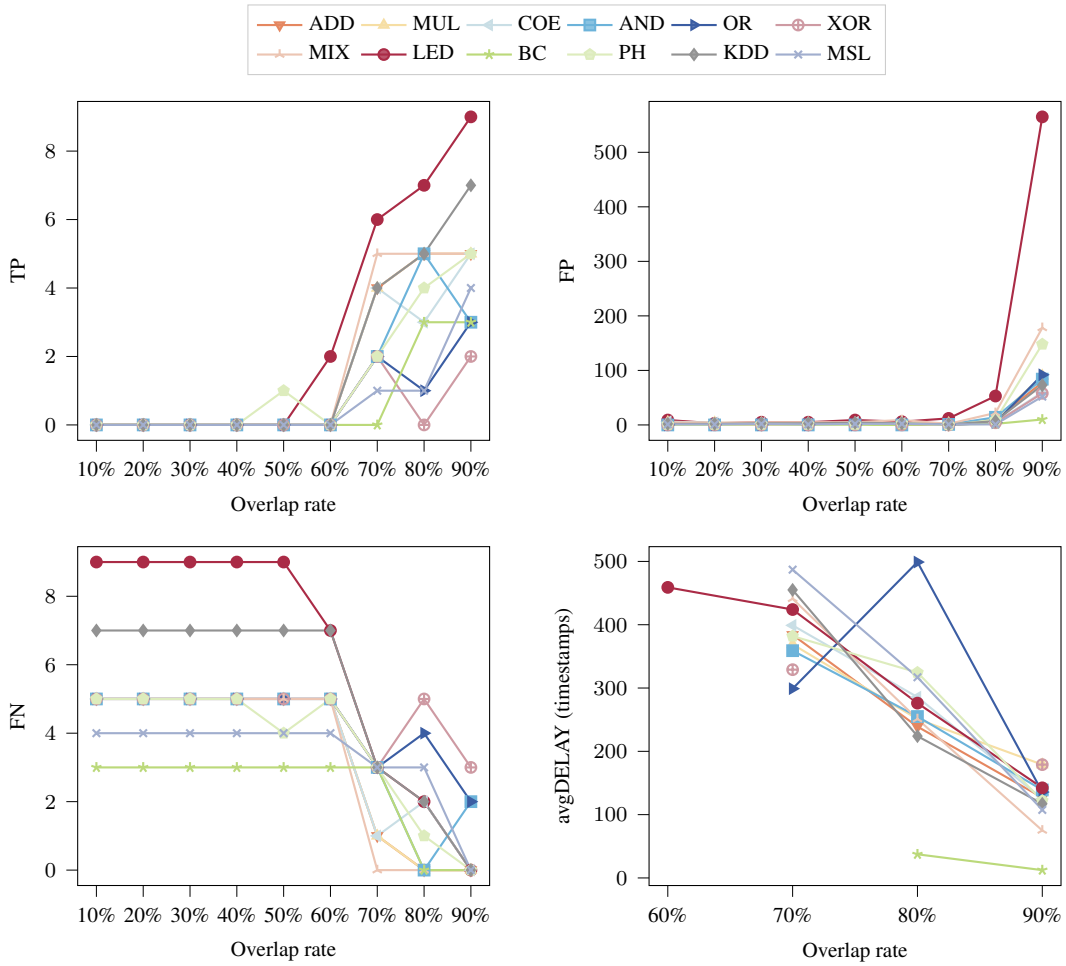


Figure 3.6: Parameter sensitivity. The impact of the overlap rate by fixed window length $d = 100$ ($d = 10$ for BC).

3.4.2.3 Parameter sensitivity

The window length d and the overlap a are the two influential parameters for constructing the sliding windows in slidSHAPs. In practice, the sliding window length can be determined empirically by the dataset's size or prior knowledge of the data (e.g., an hour, a day). Moreover, the overlap a directly impacts the generated slidSHAP representation. We conducted experiments using $\{10\%, ..., 90\%\}$ as nine different overlap rates $\frac{a}{d}$ and same window length d as in Section 3.4.2.1. Figure 3.6 shows the counts of TP, FP, FN, and the average delay for the various setups. Generally, the model detects more TP with increasing overlap rate, i.e., fine-granular slidSHAP series. However, due to the enormous increase of slidSHAP values under a

high overlap rate, the FP also explosively increases. Exploring using the different introduced datasets, we conclude that it is often reasonable to keep the overlap in the range $50\% - 80\%$; furthermore, the window length can be set up differently to detect concept drifts located with various inter-spaces among each other. Finally, the average delay fairly reflects that with a larger overlap rate, we need to conduct the statistical tests more often; therefore, it ends up with less detection delay (we do not observe average delay rates below 60% due to the absence of true positives).

3.4.2.4 KS-test versus t-test

We include both the t-test and KS-test in our experiments. In this section, we compare the two statistical tests.

Student's t-test. We restrict to the case when dealing with two separate sets of independent and identically distributed samples, thus looking for statistical significance of equality for one variable from each of the two populations. The two-sample t-test takes as the null hypothesis that the means of two populations are equal.

KS-test. The KS-test is a non-parametric and distribution-free statistical test to compare continuous one-dimensional probability distributions. It can be used to compare two sample sets in the case of the two-sample KS-test. Given two samples and their empirical cumulative distribution functions F_1 and F_2, the KS-test assumes as null hypothesis that the two samples are drawn from the same distribution; thus, given a significance level α, the null hypothesis is rejected if

$$\sup_x |F_1(x) - F_2(x)| > c(\alpha)\sqrt{\frac{m+n}{m \cdot n}} \tag{3.7}$$

where $c(\alpha) = \sqrt{-\ln(\frac{\alpha}{2}) \cdot \frac{1}{2}}$, m and n are the sizes of the two sample sets.

To compare the slidSHAPs performance under the two statistical tests, we replace the default t-test with the KS-test (significance level $\alpha = 0.05$) and refer to the new variant as slidSHAPs-KS and report the average ranking of F1 scores among competitors on all datasets. As shown in Figure 3.7, the slidSHAPs (with t-test) and slidSHAPs-KS (with KS-test) rank in first places.

3.4.2.5 slidSHAPs approximations

The computation of Shapley values is an NP-hard problem that involves evaluating the value function on each possible subset of features. The exact computation of Shapley values will soon

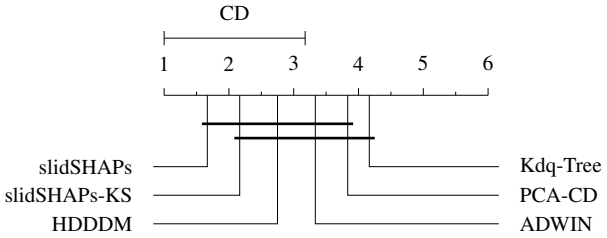


Figure 3.7: Critical difference diagram. F1 score comparison using the Nemenyi test with a 95% confidence interval (smaller numbers are better).

become unfeasible due to the exponentially growing number of evaluations involved. Several approximations appeared in the literature. Balestra et al. [BHMM22] suggest that defining an *(upper) bound* for the subsets' size on which the value function will be evaluated achieves better results than using randomly sampled subsets. We implement the same approximation for the computation of our slidSHAP series. The upper bound defined by the user influences the quality of the approximation for the Shapley values. For a N-dimensional data stream, an upper bound equals to N represents the non-approximated computation of the Shapley values. We generally use the non-approximated version of the slidSHAPs, except in the two high-dimensional real-world datasets KDD and MSL, where we use the approximated version with an upper bound equal to 2.

We conducted experiments using various upper bounds on the synthetic datasets to evaluate the runtime of the slidSHAP series' computation. Table 3.4 contains the runtimes in seconds to compute the slidSHAP series for $d = 100$ and overlap $a = 70$ when using the different upper bounds.

3.5 Conclusion

In this chapter, we proposed a new unsupervised concept drift detection approach for data streams with discrete values. slidSHAPs allows concrete visualization of concept drifts in the correlation structure of the data stream and provides a method to relocate identified drifts in the original multivariate data stream. We used data streams with input values varying in finite discrete spaces. However, slidSHAPs can be extended to real-valued streaming data using encoding techniques, e.g., [KLF05]. On the other hand, implementing a value function that supports real values (e.g., [BHMM22]) can also enable slidSHAPs for continuous data,

Table 3.4: Runtime in seconds for the computation of slidSHAP series for the synthetic datasets; columns represent different upper bounds chosen. The parameters are set to $d = 100$ and $a = 70$ for all data sets and the reported results are the averages of 10 running trials.

	2	3	4	5	6	7	8	9
ADD	50.25	63.81	68.88					
MUL	50.34	60.99	68.46					
COE	50.44	63.01	68.54					
AND	51.84	61.52	65.91					
OR	49.83	60.81	67.62					
XOR	51.00	60.56	65.87					
MIX	145.40	433.18	852.43	1344.84	1725.70	1860.64	1987.60	2046.01
LED	318.86	610.96	945.78	1223.09	1283.66			

the differential Shannon entropy for the total correlation computation can be a starting point. Furthermore, as a by-product, the slidSHAP series can be used to increase the interpretability of the detected concept drifts, using the shifts in the slidSHAPs in the neighbor of the drift positions to predict how they influence the input variable correlation in future timestamps.

Generally, the drift detection technique is the first step in the dynamic data stream anomaly detection task. Depending on the problem settings, either classical supervised drift detectors or unsupervised correlation drift detectors like slidSHAPs can be used to determine distributional drifts in data streams. The anomaly detector can then be efficiently adapted to the latest data characteristics. In the next chapter, we will introduce a strategy for online state-transition-aware modeling of anomaly detectors under concept drifts.

4

Online Adaptive Anomaly Detection

Detecting anomalies in data streams suffers from multiple challenges. On the one hand, similar to static time series data in Chapter 2, the abnormal patterns are usually hidden in the temporal context, which cannot be detected by evaluating single points. On the other hand, the normal state evolves over time due to concept drifts so that a pre-trained model easily expires. In addition to drift detection (Chapter 3), efficient model adaptation to the latest concept is also desired.

Autoencoders have recently been applied for unsupervised anomaly detection. However, they are trained on a single normal state and usually become invalid after distributional drifts in the data stream. In this chapter, we use an Autoencoder-based adaptive approach for anomaly detection under concept drifts. In particular, we propose a state-transition-aware model STAD to map different data distributions in each period of the data stream into *states*, thereby addressing the model adaptation problem in an efficient way. Our experiments evaluate the proposed method on synthetic and real-world datasets. While delivering comparable anomaly detection performance as the state-of-the-art approaches, STAD works more efficiently and provides extra interpretability.

4.1 Introduction

Anomaly detection in streaming data is gaining traction in the current big data research. Despite the high demand in a variety of real-world applications [Sip20], rare existing models show convincing performance in real-time deployment. The detection of abnormal patterns in streaming data is challenging. Labels are unavailable or expensive to acquire in real-time, such that supervised approaches usually fail. Furthermore, the conventional batch models easily expire, while a single stationary model does not fit the ever-changing data stream.

Recently, Autoencoders have been employed for anomaly detection in an unsupervised manner [MRA+16, ZSM+18]. Autoencoders are trained to reconstruct the normal data, such that for any unknown data instance, a high reconstruction error indicates an anomaly. Specifically, for time series data, the temporal dependencies between data points can be captured by constructing Autoencoders using Recurrent Neural Networks (RNNs) and their variants [MVW+15, MRA+16]. Although such methods show impressive performance on time series data, they usually ignore the fact that such data is commonly collected in a streaming way and does not allow full access during the training phase. Therefore, an adaptive Autoencoder is desired, which can be initialized with a few normal data and continuously capture the latest knowledge from the real-time data stream. Another major challenge of anomaly detection in streaming data is distinguishing between abnormal patterns and concept drifts. Once the data stream drifts to a novel distribution, a stationary model trained only on outdated data may detect most of the upcoming data undesirably as anomalies.

Given the severe problems, we aim to consider the concept drift detection and anomaly detection tasks holistically, adapt the model to the latest data distribution, and detect anomalies only concerning the temporal context where they are located. Previous concept drift detection research focuses on detecting changes of the joint probability $P(X, y)$ under a supervised setting, namely, the decision boundary changes along with the distributional changes in the input data [LLD+18]. However, for anomaly detection, the class distribution between normal and abnormal is extremely unbalanced, and labels are usually missing or delayed, so it is impractical to use traditional supervised approaches [GMCR04, BG07], e.g., detecting drifts based on the changes of real-time prediction error rate. Instead, the adaptation based on changes of the prior $P(X)$ will ensure that the Autoencoder learns the normal data pattern from the latest data distribution.

Statistical tests are commonly used for unsupervised drift detection [LLD+18]. For instance, the two-sample tests examine whether samples from two collections are generated from the same data distribution. However, many existing methods conduct tests mostly in the original input space, which are limited to linearly detectable drifts. Ceci et al. [CCJ+20] introduce both PCA and Autoencoder to embed features into a latent space for the change detection in power grid data. However, they use a feed-forward Autoencoder, which does not directly capture the temporal information in the data. A concept drift can also caused by local contextual relationship changes.

In this chapter, we propose STAD (State-Transition-aware Anomaly Detection). In STAD, data distribution in a time period is defined as a state. We use state transitions to model the concept drifts between periods. As Autoencoders are well-studied for non-linear time series anomaly detection (TSAD), we are motivated to extend the state transition paradigm to Autoencoders. We follow the standard usage of Autoencoders for anomaly detection and novelly couple the detection of concept drifts and anomalies with the informative latent representation of Autoencoders. An existing Autoencoder can be reused when a data concept reappears in the stream. A state transition is triggered by the detection of a concept drift, and this will further guide the reuse or adaptation of Autoencoders for the next period. Different from the Hidden Markov Model (HMM), which systematically models the state transition in a probabilistic manner, in our approach, we focus on state maintenance, i.e., state similarity measurement, and drift detection. Further extension w.r.t. HMM is discussed in Section 8.2

4.2 Related works

4.2.1 Data stream anomaly detection

Stationary TSAD approaches in Section 2.2.1 can be applied to data streams under the assumption that the training and testing data are uniform. However, this assumption does not hold true in common real-world applications. Hence, various online models are developed for anomaly detection. A major category of online methods is the prediction models, which employ historical data to predict the near future. Abnormal data may not fit the normal prediction and, therefore, causes a large prediction error. The widely used ARIMA model in time series analysis is also used in anomaly detection [BGBMY01]. However, using it in an online fashion requires specific adaptation strategies. The Hierarchical Temporal Memory (HTM) model [ALPA17] is designed for real-time application, while it can automatically adapt to changing statistics.

One issue with models in this category is that they are usually designed for univariate data. Therefore, deep neural networks have also recently been used to model higher dimensional and more complex data. Malhotra et al. [MVS+15] use LSTMs as a basic prediction model, which can capture the high-dimensional contextual information between different timestamps. Hundman et al. [HCL+18] also employ an LSTMs-based prediction model for anomaly detection. However, their semi-supervised approach requires partial labels from the history, which is not always possible in the streaming processing scenario.

Reconstruction-based approaches train models to reconstruct the normal data so that unknown abnormal data in the test phase will cause larger reconstruction errors due to the lack of knowledge. Autoencoders are used as an unsupervised approach for anomaly detection. Zong et al. [ZSM+18] adopt a Gaussian Mixture Model to detect anomalies from the reconstruction error. However, they use the feed-forward network, which cannot deal with inter-dependent data points as in the data stream. Malhotra et al. [MRA+16] build the Autoencoder with LSTM units to capture temporal information. Similarly, Meng et al. [MZLZ19] construct the Autoencoder with Transformers. These models assume that the sequential data are generated from the same distribution. Therefore, they are vulnerable to drifts. In the worst case, every data point that arrives after the drifts will be predicted as an anomaly.

4.2.2 Drift detection.

Recent drift detection approaches are well-summarized in [LLD+18]. Common processing paradigms aggregate the historical data, extract data features, and conduct statistical tests. Many works contribute to the streaming data classification problem [BG07, PVP18], where the real-time classification error is used as an indicator of drift detection. Unfortunately, the labels are not always immediately available in real-time. On the contrary, unsupervised drift detection methods detect changes in $P(X)$, namely the distributional changes in the streaming data. Statistical tests are usually applied to detect drifts in univariate streaming data [dRFMB16, PVP18]. For multivariate streaming data, each dimension can be tested individually and aggregated afterward [RGL19].

Finally, the model's trustworthiness and reliability are important for real-time anomaly detection, especially in safety-crucial applications. However, the interpretation of black-box anomaly detection models and complex streaming data is still under-studied. Sipple et al. [Sip20] interpret device anomalies by feature responsibility gained from Integrated Gradient [STY17].

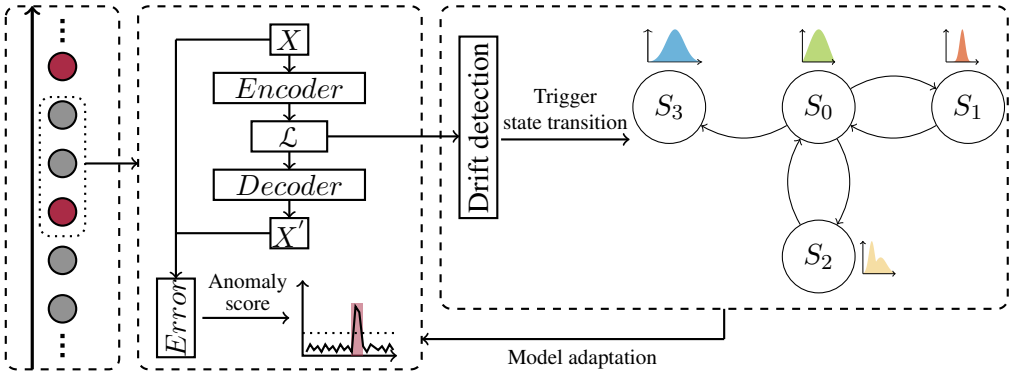


Figure 4.1: STAD overview: the left block is a multivariate data stream, where red dots denote abnormal data points and the dashed box is a data snippet. The middle block is a conventional autoencoder-based anomaly detection module, which detects abnormal snippets from the data stream. The right block takes latent representations from the autoencoder and conducts concept drift detection, which consequently triggers state transition and model adaptation.

Ahmadi et al. [AK18] uses a graph-based framework to model recurring concepts in the data stream. None of them focus on the drift detection perspective.

4.3 Methodology

In this section, we propose STAD, a state-transition-aware anomaly detection approach, which employs Autoencoders as the base model. The latent representations of Autoencoders are used to detect concept drifts, which consequently trigger state transitions. An overview of STAD is shown in Figure 4.1.

4.3.1 Preliminaries

4.3.1.1 Terminology

Data Stream and concept drift. Let $\mathcal{X} = \{X_t\}^D_{t\in\mathbb{N}}$ be a D-dimensional data stream, where X_t denotes the observation at timestamp t. The data stream contains unlabeled anomalies as well as distributional changes caused by concept drifts. Instead of explicitly categorizing different concept drift types [LLD+18], we uniformly consider that a concept drift occurs in the data stream between timestamps t and $t + c$ if the prior probability $P_{<t}(X) \neq P_{>t+c}(X)$, where $P_{<t}$ and $P_{>t+c}$ are respectively the data distribution from the last concept drift to t and from $t + c$ to the next concept drift. The period $[t, t + c]$ is the drift period, defined as the

minimum period that covers the whole distributional change. The data distribution other than drift periods is assumed to be stable. Due to the lack of labels under the unsupervised setting, we only consider the prior (virtual) shifts [LLD+18] in the data stream.

State transition. Imitating the automata theory, we formulate concept drifts in streaming data with a state transition model $\mathcal{M} = \langle \mathcal{X}, \mathcal{S}, \delta \rangle$ where $\mathcal{X}$ is a multivariate data stream, $\mathcal{S} = \{S_1, S_2, ..., S_N\}$ is a set of states (N is the user-defined maximum number of states that can be maintained), δ is a set of transition functions $\delta : \{S_i \Rightarrow S_j\}(S_i, S_j \in \mathcal{S}, i \neq j)$. For each state $S_i = \langle P_i, AE_i \rangle (i = 1, ..., N)$, AE_i is the Autoencoder trained on the current concept data, P_i is the empirically estimated distribution in the Autoencoder latent space. In this work, we assume sufficient data after the concept drifts is available to learn P_i and AE_i.

For distributional stationary data streams where no concept drift occurs, there will be only a single state without transition, and the model reduces to a single conventional Autoencoder for stationary data.

Anomaly. An observed data window $X_t^w = \{x_{t+1}, ..., x_{t+w}\}(t, w \in \mathbb{N})$ is abnormal if it significantly deviates from its temporal neighbors (data windows in the same *state*). The significance of the deviation can be determined by thresholding or statistical techniques. Both concept drifts and anomaly windows are distributionally deviating from their temporal neighbors. In our study, we distinguish them in terms of length. After the concept drifts, we assume that the data distribution stays stationary in the new concept for a significantly longer period. In contrast, after a short anomaly data window, the data stream returns to the previous distribution.

4.3.1.2 Problem statement

Given a D-dimensional data stream $\mathcal{X} = \{X_t\}_{t \in \mathbb{N}}^D$, we aim to identify any period $[t + 1, t + w]$ where the corresponding data window X_t^w is abnormal. The detection process should be unsupervised and in real-time. We also detect concept drifts in the data stream and switch to an existing Autoencoder or train a new one on the newly arrived data.

4.3.2 Reconstruction and latent representation learning

Let $f_{Enc}: \mathbb{R}^{w \times D} \rightarrow \mathbb{R}^H$ and $f_{Dec}: \mathbb{R}^H \rightarrow \mathbb{R}^{w \times D}$ be the encoder and decoder of an Autoencoder. The encoder maps a data window X_t^w of the multivariate streaming data into an H-dimensional latent representation $L \in \mathbb{R}^H$, while the decoder reconstructs the same format data window $X_t'^w$ from L, where w is the window length and $t, w \in \mathbb{N}$. A common assumption for anomaly detection using Autoencoders is that pure normal data are available for the initial

model training. The reconstruction error $e_t^w = |X_t^w - X_t'^w|$ indicates the goodness of fit to the normal data. In the test phase, abnormal data will cause larger reconstruction errors than normal data such that they are separable. The encoder and decoder can be implemented with a variety of deep models [ZSM+18, ZCLZ16]. Considering the temporal dependencies in streaming data, RNNs and their variants [MVW+15, MRA+16] are naturally suitable for the target. In the following illustration, as an example, we take the LSTM-Autoencoder [MRA+16], which takes data windows as input and produces a single latent representation for each window. To map the multivariate reconstruction error to the likelihood of anomalies, a commonly used approach is to estimate a multivariate Gaussian distribution from the reconstruction error of normal data and measure the Mahalanobis distance between the reconstruction error of an unknown data point to the estimated distribution [MRA+16]. Moreover, the Gaussian Mixture Model (GMM) [ZSM+18] and energy-based model [ZCLZ16] can also be used for likelihood estimation. The thresholding over the estimated anomaly likelihood in an unsupervised manner is challenging, especially in the real-time prediction scenario. A possible non-parametric dynamic thresholding technique is proposed in [HCL+18]. The unsupervised approach for the adaptive threshold in different periods is not our main focus. In the following sections, we focus on adapting Autoencoders based on the state transitions.

4.3.3 Drift detection in the latent space

In real-time, the latent representations of the Autoencoder are accumulated for concept drift detection. Existing concept drift detection approaches mostly work in the original space, targeting linear concept drifts. Considering the complex concept drifts in multivariate streaming data, even non-linear distributional changes can be observed in the Autoencoder latent space. We perform the non-parametric and distribution-free two-sample Kolmogorov–Smirnov test (KS-test) [CWZ+16, DKVY06] on each latent space dimension to check whether two latent representations are drawn from the same continuous distribution. Algorithm 2 shows the online concept drift detection process.

Similar to slidSHAPs in Section 3.3.3, here we also maintain two data buffers for the statistical test-based drift detection. Different from slidSHAPs, we store the latent representations in STAD. Formally, let $\mathcal{L}_{hist} = \{L_{t-\hat{m}-n+1}, L_{t-\hat{m}-n+2}, ..., L_{t-n}\}$ $(m^* \leq \hat{m} \leq m)$ be the accumulated latent representation since the last concept drift and $\mathcal{L}_{new} = \{L_{t-n+1}, L_{t-n+2}, ..., L_t\}$ be the latest latent representations. m and n are the maximum size of $\mathcal{L}_{hist}$ and $\mathcal{L}_{new}$, m^* is the minimum size of $\mathcal{L}_{hist}$ to trigger a statistical test. F_{hist} and F_{new} are the empirical estimated

Algorithm 2 Latent Space Drift Detection

Input: $\mathcal{L}_{hist}$ with maximum size m, $\mathcal{L}_{new}$ with maximum size n, minimum $\mathcal{L}_{hist}$ size m^* trigger test, current state $S = \langle P, AE \rangle$, state transition model $\mathcal{M} = \langle \mathcal{X}, \mathcal{S}, \delta \rangle$

while stream does not end **do**
  $L_t \leftarrow$ AnomalyDetection(AE,X_t^{t+w}) ▷ Get latent representation
  $\mathcal{L}_{new} \leftarrow \mathcal{L}_{new} \cup L_t$
  if $\mathcal{L}_{new}.size > n$ **then** ▷ Move the oldest element of $\mathcal{L}_{new}$ to $\mathcal{L}_{hist}$
    $L_{t-n+1} = \mathcal{L}_{new}.pop()$
    $\mathcal{L}_{hist} \leftarrow \mathcal{L}_{hist} \cup L_{t-n+1}$
  end if
  if $\mathcal{L}_{hist}.size > m$ **then**
    $\mathcal{L}_{hist}.pop()$
  end if
  if $\mathcal{L}_{hist}.size \geq m^*$ and $\mathcal{L}_{new}.size = n$ **then**
    if KSTest($\mathcal{L}^h_{hist}$,$\mathcal{L}^h_{new}$) is True **then** ▷ Equation 4.1
      $S \leftarrow$ StateTransition(S, $\mathcal{L}_{new}$, $\mathcal{S}$, δ) ▷ Section 4.3.4
      Report concept drift, clear $\mathcal{L}_{hist}$ and $\mathcal{L}_{new}$
    end if
  end if
end while

cumulative distribution functions from the two latent representation sets. The null hypothesis (i.e., the observations in $\mathcal{L}_{hist}$ and $\mathcal{L}_{new}$ are from the same distribution) will be rejected if

$$\sup_L |F_{hist}(L) - F_{new}(L)| > c(\alpha)\sqrt{\frac{\hat{m}+n}{\hat{m}\cdot n}} \tag{4.1}$$

where sup is the supremum function, α is the significance level, $c(\alpha) = \sqrt{-\ln(\frac{\alpha}{2})\cdot\frac{1}{2}}$. We maintain both $\mathcal{L}_{hist}$ and $\mathcal{L}_{new}$ as queues. m is larger than n such that $\mathcal{L}_{hist}$ contains longer and more stable historical information, while $\mathcal{L}_{new}$ captures the latest data characteristic. The drift detector will only start if $\mathcal{L}_{hist}$ contains at least m^* samples, such that the procedure starts smoothly. Since the KS-test is designed for univariate data, we conduct parallel tests in each latent dimension and report concept drift if the null hypothesis is rejected on all the dimensions. Once a concept drift is detected, we will conduct the state transition procedure for model adaptation (Section 4.3.4). The historical and latest sample sets are emptied, and we further collect samples from the new data distribution.

4.3.4 State transition model

Modeling reoccurring data distributions (e.g., seasonal changes), coupling Autoencoders with drift detection, and reusing models based on the distributional features can increase the efficiency of updating a deep model in real-time. We represent every stable data distribution (concept) and the corresponding Autoencoder as a state $S \in \mathbb{S}$. In STAD, for each period between two concept drifts in the data stream, the data distribution, as well as the corresponding Autoencoder, are represented in a queue $\mathbb{S}$ with limited size. The first state $S_0 \in \mathbb{S}$ represents the beginning period of the data stream before the first concept drift. After a concept drift, a new Autoencoder will be trained from scratch with the latest m input data windows, if no existing element in $\mathbb{S}$ fits the current data distribution. Otherwise, the state will transit to the existing one and reuse the corresponding Autoencoder. In our study, we assume that sufficient data after the concept drifts can be accumulated to initialize a new Autoencoder.

To compare the distributional similarity between the newly arrived latent representations Q and the distributions of existing states $\{P_i | i = 1, ..., N\}$, we employ the symmetrized Kullback–Leibler Divergence. The similarity between Q and an existing state distribution P_i is defined as

$$D_{KL}(P_i, Q) = \sum_{L \in \mathcal{L}} P_i(L) log \frac{P_i(L)}{Q(L)} + Q(L) log \frac{Q(L)}{P_i(L)} \tag{4.2}$$

The next step is to estimate the corresponding probability distributions from the sequence of latent representations. In [DKVY06, CWZ+16], the probability distribution of categorical data is estimated by the number of object appearances in each category. In our case, the target is to estimate the probability distribution of fixed-length real-valued latent representations. In previous research, one possibility for density estimation of streaming data is to maintain histograms of the raw data stream [SG07]. In STAD, we take advantage of the fixed-sized latent representation of Autoencoders and maintain histograms of each period in the latent space for density estimation.

Let $\mathcal{L} = \{L_1, L_2, ..., L_t\}$ be a sequence of observed latent representations, where $L_i = \langle h_1^i, h_2^i, ..., h_H^i \rangle$ and H is the latent space size, the histogram of $\mathcal{L}$ is

$$g(k) = \frac{1}{t} \sum_{L_i \in \mathcal{L}} \frac{e^{h_k^i}}{\sum_{j=1}^{H} e^{h_j^i}} \quad (k = 1...H) \tag{4.3}$$

Algorithm 3 State Transition Function

function StateTransition(S_{hist}, $\mathcal{L}_{new}$, $\mathcal{S}$, δ)
  P_{new} = DensityEstimation($\mathcal{L}_{new}$)
  if $\min_{S_i=\langle P_i, AE_i\rangle \in \mathcal{S}} \{D_{KL}(P_{new}, P_i)\} \leq \epsilon$ **then** ▷ Equation 4.4
    $\delta \leftarrow \delta \cup (S_{hist} \Rightarrow S_{min})$
    return S_{min}
  end if
  $S_{new} \leftarrow \langle P_{new}, AE_{new}\rangle$ ▷ AE_{new}: Trained on new concept data
  $\mathcal{S} \leftarrow \mathcal{S} \cup S_{new}$
  $\delta \leftarrow \delta \cup (S_{hist} \Rightarrow S_{new})$
  if $\mathcal{S}.size > N$ **then**
    Remove the oldest state and relevant transitions
  end if
  return S_{new}
end function

and the density of a given period is estimated by $P(k) = g(k)$. Hence, Equation 4.2 can be converted to

$$D_{KL}(P_i, Q) = \sum_{k=1...H} P_i(k) log \frac{P_i(k)}{Q(k)} + Q(k) log \frac{Q(k)}{P_i(k)} \tag{4.4}$$

For a newly detected concept with distribution Q, if there exists a state $S_i (i \in [1, N])$ with corresponding probability distribution P_i satisfies $D_{KL}(P_i, Q) \leq \epsilon$, where ϵ is a tolerant factor, and S_i is not the direct last state, the concept drift can be treated as a reoccurrence of the existing concept. Therefore the corresponding Autoencoder can be reused, and the state transfers to the existing state. If no Autoencoder is reusable, a new one will be trained on the latest data after concept drift. To prevent an explosion in the number of states, the state transition model $\mathcal{M} = \langle \mathcal{X}, \mathcal{S}, \delta \rangle$ only maintains the N latest states. Considering that no information about the upcoming new concept is accessible, despite a potentially high error rate, we still keep using the previous model for anomaly detection until the model adaptation is finished. In other words, the previous model is used for prediction during the upcoming *drift period*. The state transition procedure is described in Algorithm 3.

4.4 Experiments

Common TSAD benchmark datasets are often stationary without concept drift. Although some claim that their datasets contain distributional changes, the drift positions are not explicitly labeled and are hard to evaluate. To this end, we introduce multiple synthetic datasets with known positions of abnormal events and concept drifts. Furthermore, we concatenate selected real-world datasets to simulate concept drifts. We evaluate the anomaly detection performance and show the effectiveness of model adaptation based on the detected drifts. The source code of STAD is available online [1].

4.4.1 Experiment setup

4.4.1.1 Dataset description

We first generate multiple synthetic datasets from a sine and a cosine wave with anomalies and concept drifts. For initialization, we generate 5000 purely normal data points with amplitude 1, period 25 for the two wave dimensions. For the real-time testing, we generate 60000 samples containing 300 point anomalies. All synthetic datasets contain reoccurring concepts, such that we can evaluate the state-transition and model reusing of STAD. Following [PVP18], we create the drifts in three fashions, abrupt (A-$*$), gradual (G-$*$) and incremental (I-$*$). For each type of drift, we create a standard version ($*$-*easy*) and a hard version ($*$-*hard*) with more frequent drifts leaving the model less time for reaction. The drifts are created by either swapping the feature dimensions (-*Swap*-) or multiplying a factor by the amplitude (-*Ampl*-). The abrupt drifts are created by directly concatenating two concepts. The gradual drifts take place in a 2000 timestamp period, with partial instances changing to the new concepts. The incremental drifts also take 2000 timestamps, while the drift features incrementally change at every timestamp. Anomaly points are introduced by swapping the values on the two dimensions.

SMD (Server Machine Dataset) [SZN+19] is a real-world multivariate dataset containing anomalies. To simulate concept drifts, we manually compose *SMD-small* and *SMD-large*. Both only contain abrupt drifts. *SMD-small* consists of test data from *machine-1-1* to *machine-1-3*, which are concatenated in the order of *machine-1-1*⇒*machine-1-2*⇒*machine-1-1*⇒*machine-1-3*. We take each machine as a concept and *machine-1-1* appears twice. *SMD-large* consists

[1] https://github.com/KDD-OpenSource/STAD

of data from *machine-1-1* to *machine-1-8* and is composed in the same fashion with *machine-1-1* recurring after each concept. For both datasets, the train set of *machine-1-1* is used for the model initialization.

Forest (Forest CoverType) [BD99] is another widely used multivariate dataset in drift detection. To examine the performance in a real-world scenario, we do not introduce any artificial drift here, but only consider the forest cover type changes as implicit drifts. As in [DJ18], we consider the smallest class *Cottonwood/Willow* as abnormal.

4.4.1.2 Competitors

We compare our model with two commonly used unsupervised streaming anomaly detectors. The LSTM-AD [MVS+15] is a prediction-based approach. Using the near history to predict the near future, the model is less impacted by concept drifts. The prediction deviation to real values of the data stream indicates the likelihood of being abnormal. The HTM [ALPA17] model is able to detect anomalies from streaming data with concept drifts. Neither LSTM-AD nor HTM provides an interpretation of the evolving data stream besides anomaly detection.

4.4.1.3 Evaluation metric

We adopt the AUROC (Area Under the Receiver Operating Characteristic curve) score to evaluate the anomaly detection performance. An anomaly score $a \in [0, 1]$ is predicted for each timestamp. The larger a, the more likely it is to be abnormal. The labels are either 0 (normal) or 1 (anomaly). We evaluate the AUROC score over anomaly scores without applying any threshold [CZS+16] so that the performance is not impacted by the quality of the selected threshold technique.

4.4.1.4 Parameter configuration

We construct the Autoencoders with two single-layer LSTM units. All training processes are configured with a 0.2 dropout rate, $1e-5$ weight decay, $1e-4$ learning rate, and a batch size of 8. All Autoencoders are trained for 20 epochs with early stopping. We detect drifts with the KS-tests at a significance level of $\alpha = 0.05$. We restrict that $\mathcal{L}_{hist}$ has to contain at least $m^* = 50$ data point to trigger the KS-tests. We set the input window size as the sine wave period 25. For the *SMD*-based datasets, following [SZN+19], the window size is set to 100. We process the data stream using a sliding window without overlap. All experiments are conducted on an NVIDIA Quadro RTX 6000 24GB GPU and are averaged over three runs.

4.4.2 Performance

4.4.2.1 Overall performance

We compare the AUROC score in the streaming data anomaly detection task between STAD and the competitors. In STAD, we set the latent representation size $H = 50$, and the sizes of the two buffers during the online prediction phase as $m = 200$ and $n = 50$. The threshold ϵ is set to 0.0005. We evaluate the performance of STAD in each state and report the average AUROC. The results are shown in Table 4.1. STAD performs best on all synthetic datasets with abrupt and gradual drifts. In the two more complicated real-world datasets, STAD outperforms LSTM-AD and stays comparable to HTM, while requiring significantly less processing time (see Section 4.4.2.3). LSTM-AD shows a dominating performance on the two incremental datasets. Due to the fact that the value at every single timestamp changes in *I-Ampl-easy* and *I-Ampl-hard*, LSTM-AD benefits from its dynamic forecasting at every timestamp, while STAD suffers under the delay between state transitions.

Table 4.1: Anomaly detection performance (AUROC)

	STAD (Ours)	LSTM-AD	HTM
A-Swap-easy	0.986 ± 0.005	$\mathbf{0.994} \pm 0.005$	0.535 ± 0.008
A-Swap-hard	$\mathbf{0.883} \pm 0.016$	0.742 ± 0.076	0.440 ± 0.017
A-Ampl-easy	$\mathbf{0.816} \pm 0.025$	0.717 ± 0.052	0.500 ± 0.006
A-Ampl-hard	$\mathbf{0.810} \pm 0.012$	0.715 ± 0.051	0.499 ± 0.006
G-Swap-easy	$\mathbf{0.948} \pm 0.019$	0.854 ± 0.064	0.506 ± 0.008
G-Swap-hard	$\mathbf{0.926} \pm 0.030$	0.800 ± 0.082	0.502 ± 0.005
I-Ampl-easy	0.911 ± 0.014	$\mathbf{0.975} \pm 0.018$	0.488 ± 0.003
I-Ampl-hard	0.970 ± 0.017	$\mathbf{1.000} \pm 0.000$	0.470 ± 0.003
SMD-small	0.755 ± 0.067	0.562 ± 0.001	$\mathbf{0.813} \pm 0.001$
SMD-large	$\mathbf{0.763} \pm 0.016$	0.578 ± 0.002	0.762 ± 0.003
Forest	0.751 ± 0.022	$\mathbf{0.977} \pm 0.001$	0.211 ± 0.001

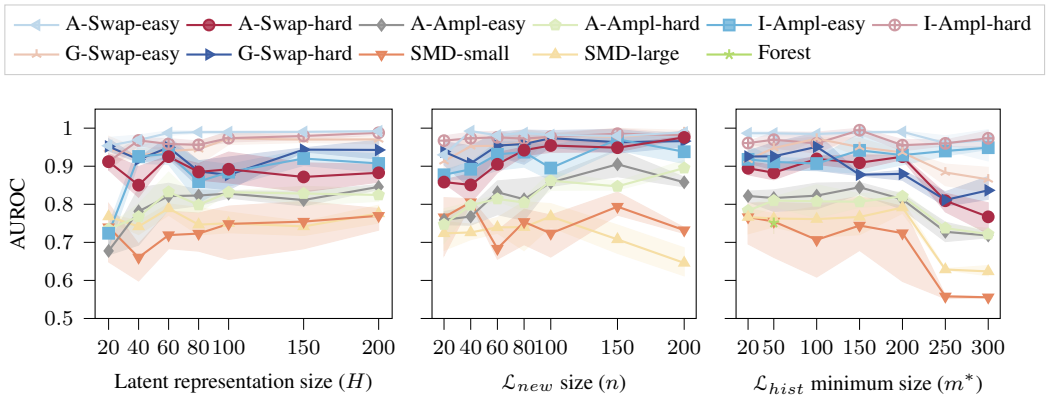


Figure 4.2: Parameter sensitivity: AUROC scores under different settings of latent representation size H, $\mathcal{L}_{new}$ size n and minimum size m^* of $\mathcal{L}_{hist}$ to trigger KS-tests.

4.4.2.2 Parameter sensitivity

In this section, we conduct multiple experiments to examine the impact of several parameters to STAD. We maintain two data buffers $\mathcal{L}_{hist}$ and $\mathcal{L}_{new}$ to collect data from the Autoencoder latent space to detect drifts. We set the upper bound of $\mathcal{L}_{hist}$'s size $m = 200$ for all experiments. Depending on the computational resource, larger m will lead to more stable test results. Here, we examine the effect of the lower bound m^*. Similarly, we also experiment with different sizes n of $\mathcal{L}_{new}$. Additionally, the latent representation size H of Autoencoders is a parameter depending on the complexity of the input data.

In Figure 4.2, we check the impact of the three parameters H, n, and m^* on abrupt drifting datasets. We try different values on each parameter while keeping the other two parameters equal to 50. The model is not sensitive to the three parameters on abrupt drifting datasets. Specifically for the two buffers, 20 data windows of both the historical (m^*) and the latest (n) latent representations are sufficient for drift detection. Similar results have been shown on the datasets with gradual and incremental drifts. The performance is stably better than that of the abrupt drifting dataset. One reason is that a longer drifting period leaves the model more time to detect the drifts and conduct the state transition. On the contrary, the model may make mistakes after an abrupt drift until sufficient data is collected and the state transition is triggered.

The other parameter ϵ controls the sensitivity of re-identifying an existing state. The larger ϵ, the more likely the model will transfer to a similar existing state. We set all H, m, and n to 50 and examine ϵ with a value that varies from 0.1 to $1e-7$ and observe the total number of distinct states created during the online prediction. As shown in Figure 4.3, with large ϵ's

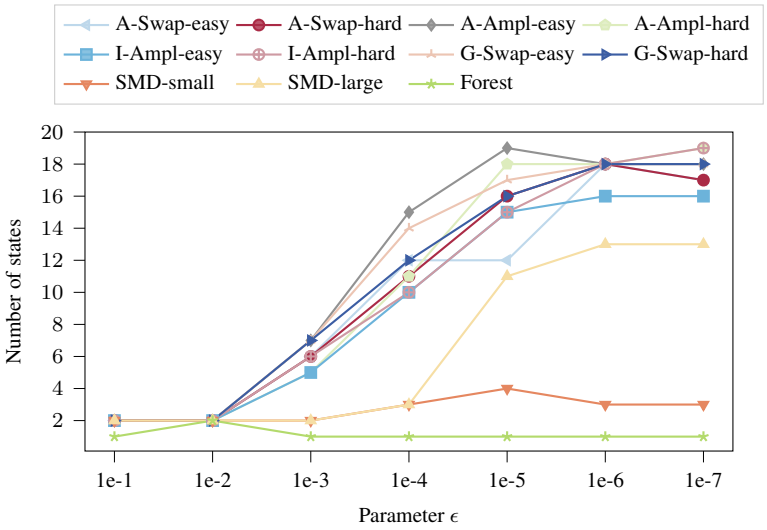


Figure 4.3: Number of distinct states under different settings of threshold ϵ.

(0.1 or 0.01), the model only creates two states and transits only between them once a drift is detected. On the contrary, too small ϵ will lead to an explosion of state. The model seldom matches an existing state but creates a new state and trains a new model after each detected drift. Currently, we determine a proper value of ϵ heuristically during the online prediction.

4.4.2.3 Running time analysis

Finally, we compare the running time (including training, prediction, and updating time) of the three models on all datasets in Figure 4.4. It turns out that the efficient reusing of existing models especially benefits large and complex datasets, where the model adaptation is time-consuming. STAD costs a similar processing time as LSTMAD in synthetic datasets and less in real-world datasets. The HTM always takes significantly more processing time.

4.5 Conclusion

In this chapter, we proposed the state-transition-aware streaming data anomaly detection approach STAD. With a reconstruction-based Autoencoder model, STAD detects abnormal patterns from data streams in an unsupervised manner. Based on the latent representation, STAD maintains states for concepts and detects drifts with a state transition model. With this, STAD can efficiently identify recurring concepts and reuse existing Autoencoders; or train a new Autoencoder when no existing model fits the new data distribution. Our empirical results

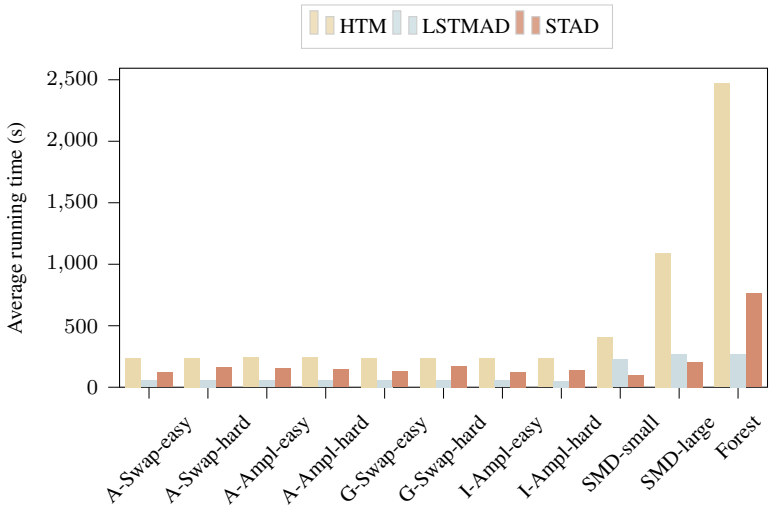


Figure 4.4: Average running time comparison.

have shown that STAD achieves comparable performance as the state-of-the-art streaming data anomaly detectors. Beyond that, the states and transitions also shed light on the complex and evolving data stream, contributing to understanding data and model changes in real-time. STAD is our first try at enabling implicit interpretability, while in Part IV further interpretation approaches will be investigated.

Part IV

Temporal Anomaly Interpretation

5

Cohesive Time Series Explanation

Beyond efficient anomaly detection in time series and data streams, we also target developing human-understandable time series explanations, which facilitates the interpretability of anomaly detection. As stated in Section 1.1.4, we aim at local, model-agnostic, and post-hoc interpretations for time series predictions. One major category of interpretation approaches is the perturbation-based models. Perturbing important features in the input data is supposed to cause a significant change in the black-box model prediction. However, perturbation-based time series interpretation suffers under two major challenges: firstly, the long and multivariate time series may lead to various incoherent salient spots on the saliency map, and secondly, common perturbation techniques often return unrealistic sequences.

In this chapter, we proposed Cohesive Explanation for Time Series (CETS). This time series interpretation approach provides cohesive (a notion of concentrated salient features at adjacent timestamps) feature attribution using realistic prototype-based perturbations. CETS ensures a cohesive interpretation by employing both global (temporal) and local (spatial) perturbations of time series. These perturbations confine the interpretation to a concise temporal event within a specific subspace. Without loss of generality, we use classification as the downstream task here. The proposed approach can be easily extended to anomaly detection. We

perform extensive experiments on real-world benchmark datasets to demonstrate the efficacy of our interpretations. Specifically, we visually illustrate how cohesive attributions contribute to enhancing the interpretability of intricate time series data. Our empirical results show that CETS achieves comparable interpretation quality to state-of-the-art approaches while providing cohesive and easy-to-understand explanations.

5.1 Introduction

With the vast development of machine learning applications in safety crucial domains [CBS+16, HWT+15], the reliability and trustworthiness of machine learning models have become increasingly prominent concerns. Despite the advancements in interpretation techniques across various deep learning domains [STY17, SGK17, RSG16], interpreting time series data and the associated sequential deep models (e.g., RNNs, GRUs, LSTMs) remains challenging. Unlike tabular and image data, the time dimension in time series makes interpretation extremely challenging. In addition to the feature importance of multivariate time series, we also emphasize understanding the temporal dynamics of feature importance throughout the entire time series. Formally, for a given time series $X \in \mathbb{R}^{N \times D}$ with N timestamps and D features, a desired interpretation should provide an importance $a(n, d)$ for each (feature $\times$ time) combination, where $n \in \{0, 1, ..., N\}$ and $d \in \{1, 2, ..., D\}$.

Recently, time series interpretation has garnered increasing attention [BSC+21, IGCBF20, LRS+23]. A major category of approaches treats time series as a 2-D image, where the two dimensions are features and timestamps, and applies classical image interpretation approaches to them [IGCBF20]. However, features and timestamps are essentially not two equivalent dimensions as in the 2-D images. The time dimension inherently carries strictly ordered temporal information, while the feature dimension does not. Consequently, classical image interpretation approaches may neglect the temporal characteristic of time series data, leading to undesired temporal feature attribution. One major problem is the *cohesiveness* of the (feature $\times$ time) attribution. Unlike saliency approaches for image data, where the salient areas with high pixel attribution can be directly visualized on images, time series data is inherently less human-understandable. Therefore, as illustrated in Figure 5.1 (left), visualizing incoherent (feature $\times$ time) attributions of time series data exacerbates the challenge, especially for long and multivariate time series. To enhance the human understandability of time series interpretations, we propose to regularize the cohesiveness in both feature and time dimensions, such that the most

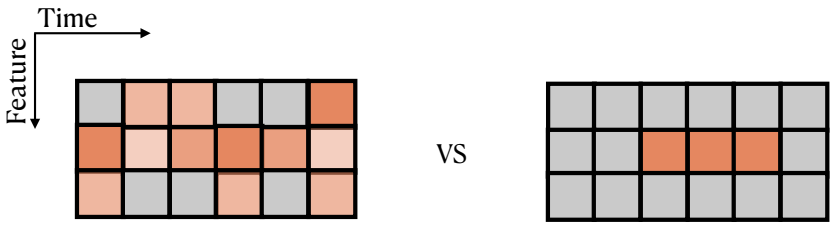


Figure 5.1: Interpretation cohesiveness: darker orange cells denotes higher importance, while gray cells denote no contribution.

salient cells are concentrated in a limited area of the saliency map, which contributes to the simplification of interpretation (Figure 5.1 right). Intuitively, the presence of a cohesive salient area indicates a brief subspace event that predominantly contributes to the model's prediction.

Perturbation-based approaches have been applied in both image and time series data. A significant prediction change after a slight perturbation of input attributes indicates high importance. Here, we address two primary challenges, the **cohesiveness** and **realisticness**. Existing perturbation-based approaches often treat a time series as a (feature $\times$ time) matrix and perturb the matrix with different strategies [IGCBF20, CS21]. However, they do not explicitly consider the cohesiveness of the perturbations, resulting in salient elements being distributed across multiple locations in the input time series matrix. A major challenge of regularizing cohesiveness is the perturbation strategy. Purely element-wise enumeration [IGCBF20] or its variants [CS21] does not consider the temporal and feature space context. Another challenge in perturbing time series data is the realisticness of perturbed sequences. For example, replacing a sub-sequence of an ECG snippet with random noise or temporal neighbor values could potentially compromise the semantics of the input data.

To this end, we propose CETS, a perturbation-based time series interpretation approach. CETS aims at perturbations on *subspace events*. Specifically, we introduce a two-stage global-local (temporal-spatial) perturbation. We perturb a consecutive sub-sequence as an event and perturb a sub-feature space to find the most influential features of that event. CETS utilizes *prototype* for the subspace event perturbation. Prototypes are representative time series of the training data, e.g., centers of time series clusters. The prototypes are considered as unified and less informative sequences. Using the corresponding prototype to perturb time series subspace events not only contributes to the cohesiveness of perturbation but also alleviates unrealistic perturbations [LJM23].

The main contribution of this chapter can be summarized as follows:

1. **two-stage perturbation-based approach**: we introduce a novel two-stage perturbation-based method for interpreting time series data, with a primary focus on ensuring the cohesiveness of attributions

2. **prototype utilization for realistic perturbation**: to achieve realistic subspace event perturbation, we employ prototypes, which are representative existing time series, alleviating unrealistic perturbations

3. **empirical validation**: we substantiate the effectiveness and interpretability of our proposed approach through extensive experiments conducted on well-established real-world benchmark datasets.

5.2 Related works

5.2.1 Perturbation-based interpretation

Perturbation-based interpretation is a local and model-agnostic approach, which has been applied in multiple application areas [ZF14, RSG16]. Regardless of the data format, the attribution is indicated by the prediction difference between original and perturbed inputs. Different strategies can be applied to create the perturbations. Feature Ablation (FA) replaces the input features with a predefined baseline. Fisher et al. [FRD19] split the data into two subsets and use the feature values in one subset to perturb the other. Feature Occlusion (FO) [ZF14] restricts the perturbed features to contiguous regions in image data. In computer vision tasks, the perturbation of an image can be setting pixel intensity to zero [RSG16, ZF14], blurring the area [FV17] or applying a random combination of regional pixels [PDS18].

However, due to the temporal nature, perturbing time series data with classical strategies may lead to suboptimal results [SAEA+19]. Ismail et al. [IGCBF20] use a two-stage approach, which first perturb timestamps with zeros and then perturb features at important timestamps with zeros. Schlegel et al. [SOKEA20, SK23] consider the temporal information in time series data and define the perturbation on continuous intervals. The original values in the selected interval are replaced by zeros, the inverse, or their mean value. Nevertheless, the perturbed sequences are not ensured with realisticness and validity, sometimes the perturbed sequence is even out-of-distribution.

Indeed, for time series data, determining the perturbation interval and the replacement values is challenging. One solution is to learn perturbation implicitly. Mask-based perturbation learns masks over the input time series to hide informative features. Mask-based approaches are first applied in computer vision tasks [FV17, FPV19] and then extended to time series domain [CS21]. Gaussian blur is commonly used as a replacement for the masked positions. Enguehard [Eng23] further improves the mask-based approaches with learnable replacement values. Pan et al. [PHC21a] employ a learnable binary mask time time series forecasting interpretation.

5.2.2 Time series prediction interpretation

In addition to the perturbation-based interpretation approaches, several other types of time series interpretation approaches have been recently developed. One category is the Shapley-value-based based approach. Unlike the perturbation-based approaches, which analyze the model performance decrease, Shapley-value-based approaches are based on the magnitude of feature attributions [Mol22]. SHAP [LL17] generates contrastive feature attributions based on the game-theoretic Shapley values. Shapley Value Sampling (SVS) [MCFH22] conducts random permutations to the input values. However, the large search space makes Shapley-value-based approaches naturally computationally expensive. This problem becomes even more severe if we consider the time dimension of the time series as an additional set of features. Bento et al. [BSC+21] extends SHAP to sequential data by considering a time series as a (feature $\times$ time) matrix. By introducing a pruning method over timestamps, TimeSHAP [BSC+21] also achieves computational effectiveness.

Probabilistic models are also used in time series prediction interpretation. FIT [TJC+20] evaluates the feature importance by quantifying the shift of predictive distribution over time using KL-divergence. They compare the distributional shift after turning off a feature at a timestamp. WinIT [LRS+23] further improves this approach by explicitly considering the distributional shift caused by multiple interdependent timestamps.

The attention mechanism has also been used for time series interpretation. The general idea is to treat the attention scores as the importance of input data. RETAIN [CBS+16] adds an attention layer to the conventional RNNs and observes the attention score to timestamps and features in the hidden space. Similarly, Karim et al. [KMDC17] integrate the attention mechanism in LSTMs and CNNs. Vinayavekhin et al. [VCM+18] directly feed the input to a temporal context layer without recurrent units.

5.2.3 Explanation cohesiveness

Despite the development of interpretation approaches, the quality of interpretation still needs to be improved to access and quantify. Nauta et al. [NTP+23] emphasized quality quantification in a human-oriented subjective perspective. In Chapter 6, we will talk about the explanation consistency [BLM23], namely, the feature attribution should be consistent between adjacent sliding windows. Different from that, here we consider the explanation consistency within a specific sliding window, targeting cohesive areas in the saliency map. Crabbe et al. [CS21] use a similar idea to simplify the learned perturbation masks in order to restrict the salient elements for the attribution result.

5.2.4 Prototype-based explanation

Prototypes [LLCR18] are example-based explanations that provide explicit and intuitive interpretation using representative data instances. Especially for complex data like time series, prototypes help to understand the data. Li et al. [LLCR18] and Gee et al. [GGOGP19] use Autoencoders to learn prototypes in the latent space. Similarly, Li et al. [LJM23] use prototypes to explain anomalies in time series. More details about prototype-based explanation are introduced in Chapter 7. Unlike previous works, we use prototypes not directly as time series interpretations but as realistic perturbations to analyze feature attribution.

5.3 Methodology

In this section, we introduce the proposed CETS model. Firstly, we formally define our notations and the problem in Section 5.3.1. Followed by an overview of the model architecture in Section 5.3.2. The subsequent subsections delve into a detailed exposition of each component comprising CETS.

5.3.1 Preliminaries

Let $X \in \mathcal{X}(\mathcal{X} \subset \mathbb{R}^{N \times D})$ be a $D-$dimensional time series with length N and $S = \{1, 2, ..., D\}$ be the feature space. And let $Y \in [1, 2, ..., K]$ be the class label of a sequence X in a classification task with K classes. We aim to explain a black-box model f using a perturbation-based approach.

To ensure a temporally cohesive explanation, we leverage prototype-based perturbations. Thus, the perturbations are generated by consecutive realistic prototypical snippets of the time series data. Let $\{p_1, p_2, ..., p_M\}$ denote a set of M distinct prototypes of $\mathcal{X}$ with $p_i \in \mathbb{R}^{N \times D}$ for $i \in \{1, 2, ..., M\}$. We further define *temporal* and *spatial* perturbations of X using a prototype $p \in \{p_1, p_2, ..., p_M\}$.

Definition 3 (Temporal perturbation)**.** The temporal perturbation on time period $\{i+1, \ldots, i+L\}$ with L timestamps replaces all features of X with the corresponding features of the most similar prototype p in the same time period. Formally,

$$\pi_t(X, i, L, p) = [X_{1,i}; p_{i+1,i+L}; X_{i+L+1,N}] \tag{5.1}$$

Definition 4 (Spatial perturbation)**.** The spatial perturbation on the feature set $T \subseteq S$ replaces values on all timestamps of X by the corresponding values of the most similar prototype p. Formally, for $d \in S$,

$$\pi_s^d(X, T, p) = \begin{cases} p^d & \text{if } d \in T \\ X^d & \text{else} \end{cases} \tag{5.2}$$

Remark 1. We restrict the temporal and spatial perturbation operation on consecutive timestamps so that the final interpretation will be cohesive sequences on the (feature $\times$ time) saliency map.

Lemma 1. *The temporal and spatial perturbations can be combined such that the time series X is perturbed by prototype p on the feature space $T \subseteq S$ for the period $\{i+1, \ldots, i+L\}$. Formally, for $d \in S$,*

$$\pi_{ts}^d(X, i, L, T, p) = \begin{cases} [X_{1,i}^d; p_{i+1,i+L}^d; X_{i+L+1,N}^d] & \textit{if } d \in T \\ X^d & \textit{else} \end{cases} \tag{5.3}$$

Remark 2. Due to the lack of supervision on feature importance generally in time series classification benchmark datasets, we use the prediction difference between $\hat{y}^* = f(X)$ and $\hat{y} = f(\pi_{(.)}(X))$ to implicitly signify the feature importance of perturbed elements.

5.3.2 Architecture overview

An overview of the CETS architecture is visualized in Figure 5.2. In the model initialization phase, a labeled train set is utilized to train a black-box model for classification and learn a set of prototypes. More details about prototype learning are described in Section 5.3.3. After the

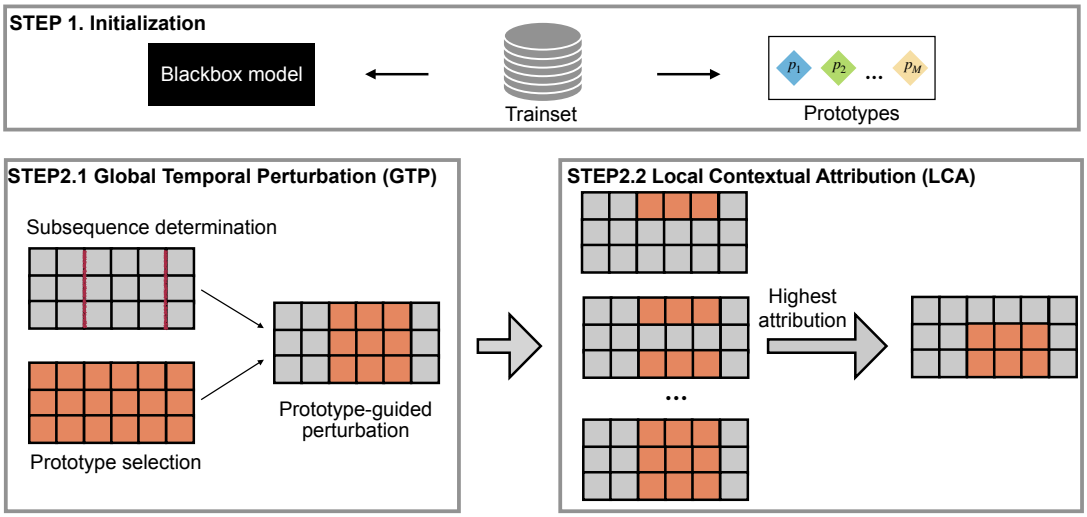


Figure 5.2: Architecture overview.

model's prediction on test data, the prototypes will be used for the perturbation-based interpretation. A two-stage global-local perturbation strategy (Sections 5.3.4 and 5.3.5) will be applied on each time series sequence. The ultimate temporal feature attribution will be determined by assessing the prediction change after the prototype-based perturbation. The procedural steps of CETS are elucidated in Algorithm 4.

5.3.3 Time series prototype learning

In CETS, we learn the prototypical sequence from the training data, which will be used for the permutations in the follow-up explanation steps. We adopt a K-Medoid-based approach [KR09], where the cluster medoids are regarded as prototypes. Considering the temporal characteristic of time series data, we employ Dynamic Time Warping (DTW) [RK05] to quantify the dissimilarity between time series. In K-Medoid, the cluster representatives (medoids) are selected from real training data and updated by swapping between medoid and non-medoid points, such that the learned prototypes are ensured to be existing time series sequences. With this, we prevent generating unrealistic prototypes [LJM23, GGOGP19], which could subsequently affect the perturbation quality.

We define M as the number of clusters, namely the number of prototypes to be learned.

$$\{p_1, p_2, ..., p_M\} = KMedoids(\mathcal{X}_{train}, M) \tag{5.4}$$

Algorithm 4 Cohesive Explanation for Time Series (CETS)

Input: time series $\mathcal{X} \subset \mathbb{R}^{N \times D}$, perturbation length L, number of prototypes M, black-box model f

$\mathcal{A} = [\ ]$
$\mathcal{X}_{train}, \mathcal{X}_{test} \leftarrow train\_test\_split(\mathcal{X})$
for $c \in \mathcal{X}.classes$ **do**
    $P^c \leftarrow KMedoids(\mathcal{X}^c_{train}, M)$ ▷ Equation (5.4)
end for
for $X \in \mathcal{X}_{test}$ **do**
    $\hat{c} \leftarrow f.predict(X)$
    $p^* \leftarrow \underset{p_i \in P^{\hat{c}}}{\operatorname{argmin}}(dist(X, p_i))$
    $i^* \leftarrow \underset{i \in \{1,\dots,N-L\}}{\operatorname{argmin}} GTP(X, i, L, p^*)$ ▷ Algorithm 5
    $T^* \leftarrow \underset{T \subseteq S}{\operatorname{argmin}}\ LCA(X, i^*, L, T, p^*)$ ▷ Algorithm 6
    $a^* \leftarrow a(i^*, T^*)$ ▷ Equation (5.7)
    $\mathcal{A}.add(a^*)$
end for
return $\mathcal{A}$

Typically, we conduct the K-Medoids algorithm for each class in the train set to represent all classes equally. The number of clusters M is a hyperparameter. During the interpretation phase, we consider the class prediction of the black-box model to be correct and use the predicted class to find the corresponding set of prototypes. For a given time series sequence undergoing explanation, the *most similar prototype* is determined by clustering the sequence to one of the K-Medoids clusters of the predicted class. The selected prototype will be used in Section 5.3.4 and Section 5.3.5 for perturbation.

5.3.4 Global temporal perturbation

The global temporal perturbation (GTP) conducts a temporal perturbation $\pi_t(.)$ on a given time period of X, which aims to locate the most important period $\{i+1, \dots, i+L\}$ within a time series sequence.

$$GTP(X, i, L, p) \coloneqq \pi_t(X, i, L, p) \tag{5.5}$$

For a specific globally perturbed time series $\tilde{X}_{GTP} = GTP(X, i, L, p)$ w.r.t. time period

Algorithm 5 Global Temporal Perturbation (GTP)

Input: time series $X \in \mathbb{R}^{N \times D}$, most similar prototype p, perturbation length L

1: **for** $i \in \{1, ..., N - L\}$ **do**
2: $\quad \tilde{X}_{GTP} \leftarrow \pi_t(X, i, L, p)$ ▷ Equation (5.5)
3: $\quad a(i) \leftarrow |f(X) - f(\tilde{X}_{GTP})|$
4: **end for**
5: **return** $\underset{i}{\text{argmax}}\{a(i)\}$

starts at i and last a fix length L, the attribution is defined by $a(i) = |\hat{y}^* - \hat{y}(i)|$, where $\hat{y}^* = f(X)$ being the prediction of original time series and $\hat{y} = f(\tilde{X}_{GTP})$ being the prediction of X applied GTP starting from timestamp i.

GTP distinguishes temporal importance by always considering the whole feature space. The intuition is to capture global features of time series in this period, e.g., trend, seasonality, periodicity. The most important time period starting from timestamp i is determined by $i = \underset{i}{\text{argmax}}\{a(i)\}$ for $i \in \{1, ..., N - L\}$. The time period $\{i, \ldots, i + L\}$ will be further used in the next step for a more detailed local attribution analysis. The entire process of GTP is presented in Algorithm 5.

5.3.5 Local contextual attribution

The local contextual attribution (LCA) focuses on the time period $\{i, \ldots i + L\}$ determined by GTP and treats it as a local context for local attribution analysis. Specifically, LCA conducts a spatial perturbation $\pi_{ts}(.)$ on feature set $T \subseteq S$ for the most important time period $\{i, \ldots, i + L\}$,

$$LCA(X, i, L, T, p) \coloneqq \pi_{ts}(X, i, L, T, p) \tag{5.6}$$

for $\pi_{ts} = [\pi_{ts}^1, \pi_{ts}^2, \ldots, \pi_{ts}^D]$.

Given the most important time period $\{i, \ldots, i + L\}$ from GTP, for a specific locally perturbed time series $\tilde{X}_{LCA} = LCA(X, i, L, T, p)$ w.r.t. feature subset $T \subseteq S$, the attribution is defined by

$$a(i, T) = \frac{1}{\sqrt{|T|}} |\hat{y}^* - \hat{y}(i, T)| \tag{5.7}$$

where $\hat{y}(i, T) = f(\tilde{X}_{LCA})$ being the prediction of X applied LCA on T and the score is normalized by the $\sqrt{|T|}$, in order to eliminate the impact of the numeral difference caused by different subspace sizes.

Algorithm 6 Local Contextual Attribution (LCA)

Input: time series $X \in \mathbb{R}^{N \times D}$, most similar prototype p, perturbation start time point i, perturbation length L

▷ S is the feature space of X with ($|S| = D$)

1: **for** $T \in \mathcal{P}(S) \backslash \emptyset$ **do**
2: $\quad \tilde{X}_{LCA} \leftarrow \pi_{ts}(X, i, L, T, p)$ ▷ Equation (5.6)
3: $\quad a(i, T) \leftarrow \frac{1}{\sqrt{|T|}} |f(X) - f(\tilde{X}_{LCA})|$
4: **end for**
5: **return** $\underset{T}{\operatorname{argmax}}\{a(i, T)\}$

For a D-dimensional time series data, we examine all the 2^{D-1} non-empty subsets of the feature space to evaluate the most important subspace for LCA. The most important subspace is $T = \underset{T}{\operatorname{argmax}}\{a(i, T)\}$ for $T \in \mathcal{P}(S) \backslash \{\emptyset\}$ where $\mathcal{P}$ denotes the power set. The process of LCA is presented in Algorithm 6.

The final perturbation of time series X is $\tilde{X}_{LCA}$, with values on feature subset T between timestamps i and $i+L$ being replaced by the corresponding values in the most similar prototype p.

5.4 Experiments

5.4.1 Experimental setup

5.4.1.1 Dataset description

We empirically evaluate the interpretation of CETS on multiple real-world time series classification benchmark datasets. We select both univariate and multivariate time series data from the UCR Time Series Classification Archive [DBK+19]. A summary of the datasets is provided in Table 5.1.

5.4.1.2 Competitors

We compare the CETS interpretation with multiple other methods from different categories. Firstly, we compare with the gradient-based methods DeepLIFT [SGK17], GradientSHAP (GradSHAP) [EJS+19] and Integrated Gradien (IG) [STY17]. Furthermore, we also include Feature Occlusion (FO) [ZF14] and Augmented Feature Occlusion (AFO) [TJC+20] from

Table 5.1: Dataset description

Dataset	Length	Dimensionality	Classes
Meat (MEA)	448	1	3
GunPointAgeSpan (GUN)	150	1	2
SonyAIBORobotSurface2 (SON)	65	1	2
EthanolConcentration (ETH)	1751	3	4
SelfRegulationSCP1 (SCP)	896	6	2
RacketSports (RAC)	30	6	4

the feature perturbation-based approaches, and DynaMask [CS21] from the mask-based approaches. Finally, we compare with WinIT [LRS+23] as the probabilistic approach.

5.4.1.3 Evaluation metric

To quantify the quality of interpretation in an unsupervised fashion, we use the performance drop in AUROC (Area Under the Receiver Operating Characteristic curve). To be noticed, instead of beating all the competitors, we primarily aim to deliver cohesive time series interpretation while staying comparable in quality to other competitors. Therefore, we also focus on the ranking of the AUROC score drop.

5.4.1.4 Parameter configuration

We implement the black-box time series predictor as a single-layer GRU with 200 hidden units, followed by a fully connected linear output layer. The model is trained over 200 epochs with early stopping using the Adam optimizer [KB14] with 10^{-3} learning rate and 0.5 dropout rate. The experiment results are averaged over five runs.

For CETS, we uniformly learn 5 prototypes for each class in the training data. By default, we set perturbation length $L = \frac{N}{10}$, namely 10% of the time series length N. We also provide a parameter sensitivity analysis in Section 5.4.2.3. The source code of CETS is available online [1].

[1] https://github.com/KDD-OpenSource/CETS

5.4.1.5 Masking strategy

In contrast to point-wise-ranked attribution approaches that involve masking the subsequential timestamps of the top-ranked (feature$\times$ time) elements [LRS+23], CETS utilizes prototypes to mask consecutive time periods in order to achieve the optimal cohesiveness in the generated attribution explanation. Specifically, in CETS, we directly perturb the time period on the most important features using the output of the LCA process. To make a fair comparison, we also implement a prototype-based perturbation version of the competitor approaches using their own attributions. For the competitor models' attribution of the time series X, we identify the highest element (i, d), where i denotes the i-th timestamp, and d represents the d-th feature. For a prototype-based perturbation with perturbation length L, we take the most similar prototype p of X, and perturb X with

$$\tilde{X} = \begin{cases} [X_{1,i}; \tilde{p}; X_{i+L+1,N}] & \text{if } i \leq N - L \\ [X_{1:N-L}; \tilde{p}] & \text{else} \end{cases} \tag{5.8}$$

where

$$\tilde{p} = \begin{cases} [X^{1,d-1}_{i+1,i+L}; p^{d}_{i+1,i+L}; X^{d+1,D}_{i+1,i+L}] & \text{if } i \leq N - L \\ [X^{1,d-1}_{N-L+1,N}; p^{d}_{N-L+1,N}; X^{d+1,D}_{N-L+1,N}] & \text{else} \end{cases} \tag{5.9}$$

5.4.2 Performance

5.4.2.1 Interpretation evaluation

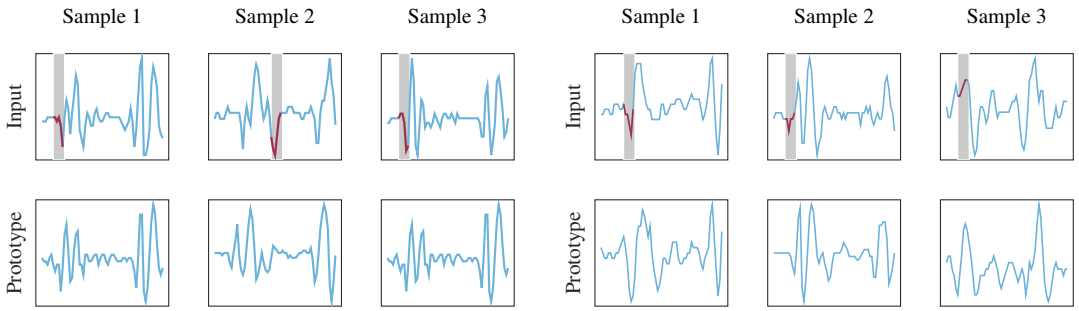
Following the masking strategy in Section 5.4.1.5, we quantify the performance drop of compared approaches and report the average ranking among the eight interpreters in Table 5.2. Smaller numbers indicate higher ranking, i.e., averagely larger performance drop by perturbation. CETS achieves the best average ranking on GUN, SON, ETH, SCP, and second best for other datasets. To be noticed, ETH always ranked last for WinIT because WinIT does not converge during feature generation [LRS+23] on ETH. In summary, CETS provides outstanding feature attribution quality w.r.t. performance drop analysis. Beyond this, CETS enables better cohesive interpretability using prototype-based perturbations.

5.4.2.2 Cohesive interpretation

A significant advantage of CETS is that the prototypes employed for perturbation also contribute to the cohesive interpretation. The prototypes, corresponding to instances with the largest performance drop, inherently indicate the most crucial time span and features in the

Table 5.2: AUROC drop ranking (Best: bold, second best: underline)

	MEA	GUN	SON	ETH	SCP	RAC
IG	6.200±1.304	6.600±1.140	6.800±0.447	6.000±1.225	2.800±2.387	4.600±2.510
FO	4.800±2.280	2.600±1.673	3.600±1.140	4.800±1.483	6.000±1.732	3.800±2.588
AFO	5.600±1.342	5.600±1.949	2.600±1.140	3.000±1.414	5.400±0.894	4.800±1.483
GradSHAP	4.800±1.304	6.000±1.581	4.600±1.140	6.000±0.707	4.000±2.550	5.400±2.074
DeepLIFT	6.400±1.517	2.400±1.140	8.000±0.000	4.000±2.121	4.600±1.140	5.600±2.510
WinIT	4.200±3.493	4.400±2.191	2.600±1.517	8.000±0.000	7.200±1.304	6.400±1.673
DynaMask	**1.600±0.894**	6.600±0.894	6.000±0.707	2.400±1.342	3.800±2.775	**2.000±1.414**
CETS (Ours)	2.400±0.894	**1.800±0.837**	**1.800±1.304**	**1.800±1.095**	**2.200±1.304**	3.400±2.191

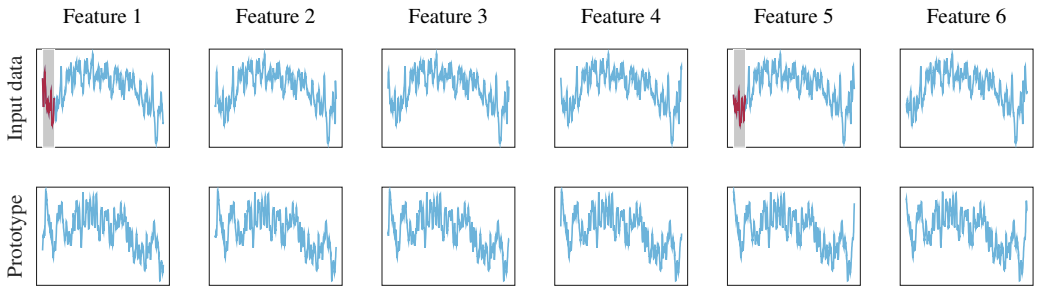


(a) Class 1 **(b)** Class 2

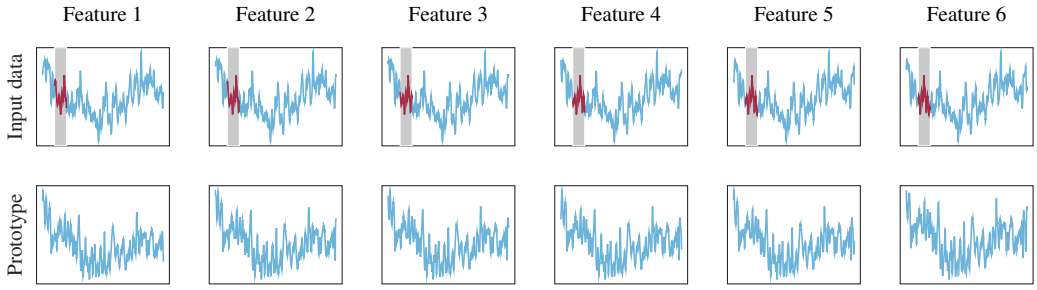
Figure 5.3: CETS time series prediction interpretation: SON dataset.

original data. We visually present several sample pairs of the raw data and their most similar prototypes from both the univariate SON and the 6-dimensional SCP data. In Figure 5.3, we showcase three samples from each class of SON, where the highlighted snippet in the input data represents the period with the highest attributing importance according to CETS. Perturbing this highlighted snippet by the corresponding prototype leads to the most substantial performance drop. Due to the cohesive perturbation in CETS, only a consecutive snippet is highlighted for each input time series, directly emphasizing the most relevant features for the prediction.

Similarly, in Figure 5.4 we also visualize the output of CETS for both the "Negativity" and "Positivity" classes from SCP. For this multivariate data, the advantages of employing CETS for subspace feature importance interpretation become more apparent. After the most relevant time period is determined by GTP (Algorithm 5), the most relevant feature combination is learned by LCA (Algorithm 6). For the "Negativity" class, the period at the beginning of the



(a) Class "Negativity"



(b) Class "Positivity"

Figure 5.4: CETS time series prediction interpretation: SCP dataset.

selected sample on feature 1 and 5 are determined with the highest contribution. And for the "positivity" class sample, a period of the full feature space is determined.

5.4.2.3 Parameter sensitivity analysis

The perturbation length L determines the number of consecutive timestamps to be perturbed by prototypes in GTP (Algorithm 5). To assess its sensitivity, we conduct an analysis with different perturbation rates α, where $L = \lfloor \alpha \cdot N \rfloor$. Specifically, we test values of α within the set $\{1\%, 5\%, 10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%\}$. The results are presented in Figure 5.5. Generally, CETS is insensitive to the perturbation rate on the tested datasets. A slightly increasing trend with increasing perturbation rate indicates that perturbing the time series with less informative prototypes leads to a performance drop. In addition, we also examined different numbers of prototypes per class M, from 3 to 30. The result in Figure 5.6 shows general stability of performance drop when increasing the number of prototypes. Even three per class is sufficient for the perturbation.

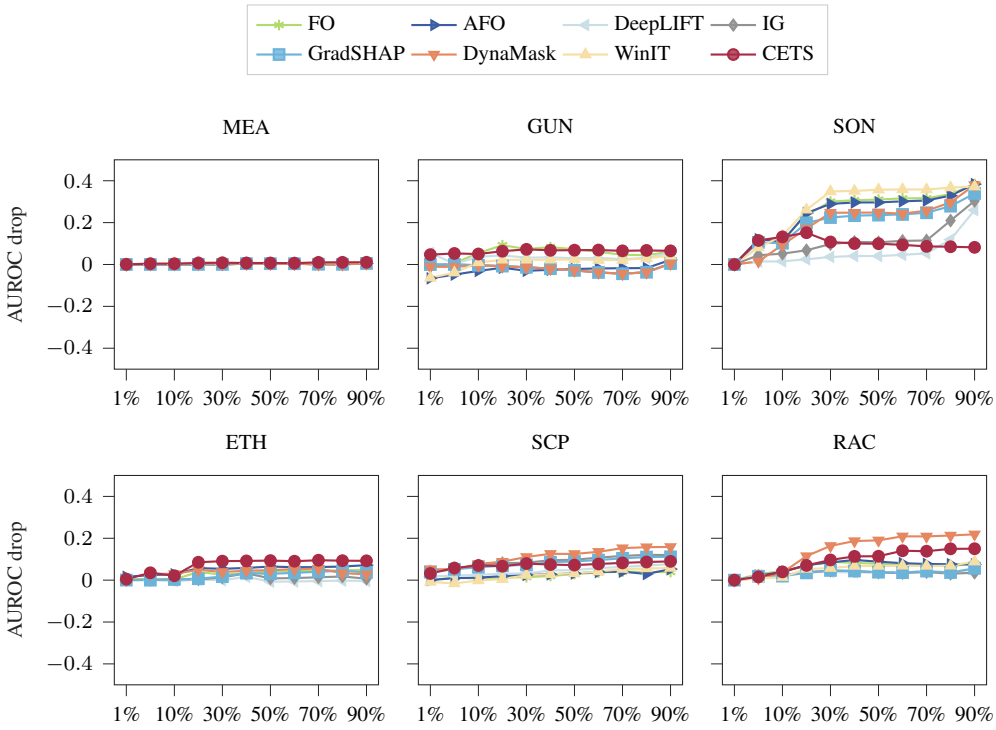


Figure 5.5: Perturbation rate sensitivity analysis.

Table 5.3: Ablation study: AUROC performance drop

	MEA	GUN	SON	ETH	SCP	RAC
w/o GTP	0.000±0.000	0.006±0.003	0.002±0.002	0.001±0.000	0.001±0.000	0.023±0.007
w/o LCA	**0.001±0.001**	**0.052±0.034**	**0.228±0.065**	0.000±0.000	0.000±0.000	0.005±0.007
CETS	**0.001±0.001**	**0.052±0.034**	**0.228±0.065**	**0.002±0.001**	**0.002±0.001**	**0.066±0.031**

5.4.2.4 Ablation study

To evaluate the contribution of the two major building blocks GTP and LCA in CETS, we conduct an ablation study to remove one component at a time and compare with the complete CETS. Due to the significant difference in number of masked elements between configurations, in Table 5.3, we report the rescaled AUROC drop $\gamma^* = \frac{\gamma * 10}{|Mask|}$, where γ and γ^* are the AUROC drops before and after rescaling and $|Mask|$ is the number of masked elements of a configuration. For the univariate datasets, we only use GTP to find the most important time period for the single feature, therefore, the performance of "w/o LCA" is the same as CETS. In general, we observe that CETS, incorporating both GTP and LCA, achieves the best performance.

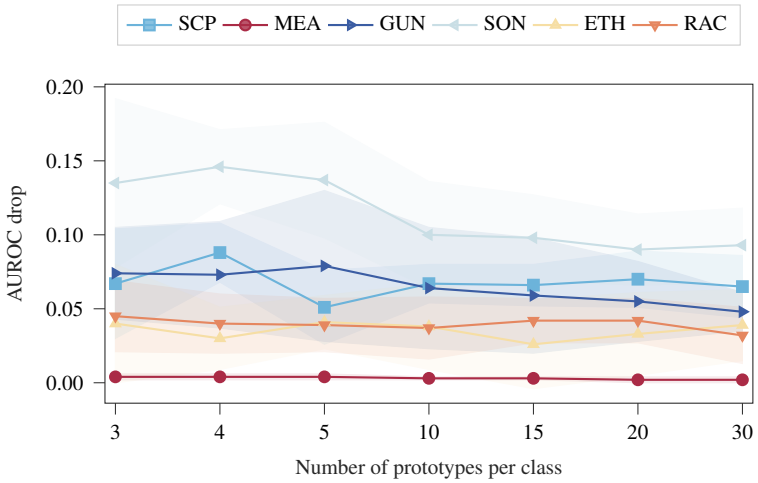


Figure 5.6: Number of prototypes sensitivity analysis.

5.5 Conclusion

In this chapter, we presented CETS, a perturbation-based time series interpretation approach using prototypes. Specifically, we achieve the cohesive explanation by conducting a two-step global-local perturbation to acquire consecutive attributions. Our experiments show that CETS achieves comparable interpretation performance to state-of-the-art time series interpretation approaches. Moreover, CETS additionally guarantees the cohesiveness of interpretation. Current prototype-based perturbation is based on sliding window (GTP) and exhaustive subspace search (LCA). Efficient search strategies for both time and feature domains are promising extension possibilities.

So far, we have examined the cohesiveness of time series saliency maps. Actually, there are still many aspects that new interpretation approaches should consider. In Chapter 6, we showcase two of those important factors, consistency, and robustness, which are defined over multiple adjacent sliding windows.

6

Consistency and Robustness

Apart from cohesiveness in a single time series sliding window, more properties are also desired to form a good qualified time series interpretation. This chapter demonstrates the *consistency* and *robustness* between adjacent sliding windows as two additional important properties. Specifically, we examine saliency explanations from both perturbation-based and gradient-based explanation models in a time series classification task. Our experimental results on five real-world datasets show that they all lack consistent and robust performances to some extent. We emphasize the importance of developing consistent and robust explanations for time series prediction by drawing attention to the flawed saliency explanation models.

6.1 Introduction

Saliency approaches for time series explanation confront essential challenges due to the complex temporal structure in time series data. The temporal dependency leads to time-dependent changes in feature attribution. Furthermore, explanation approaches are often not directly applicable to time series models with recurrent- or attention-based components [CBS+16, JW19]. Although several approaches try to treat data windows of time series as images and apply vision

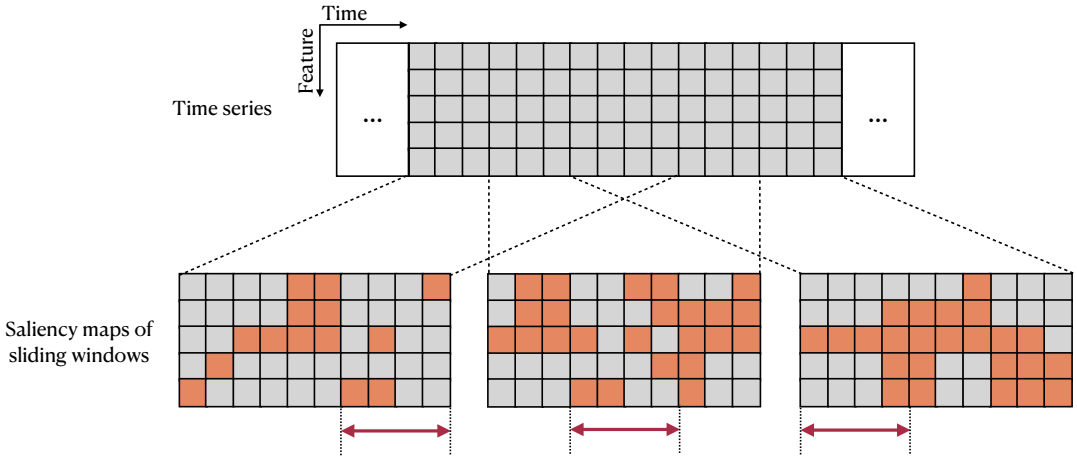
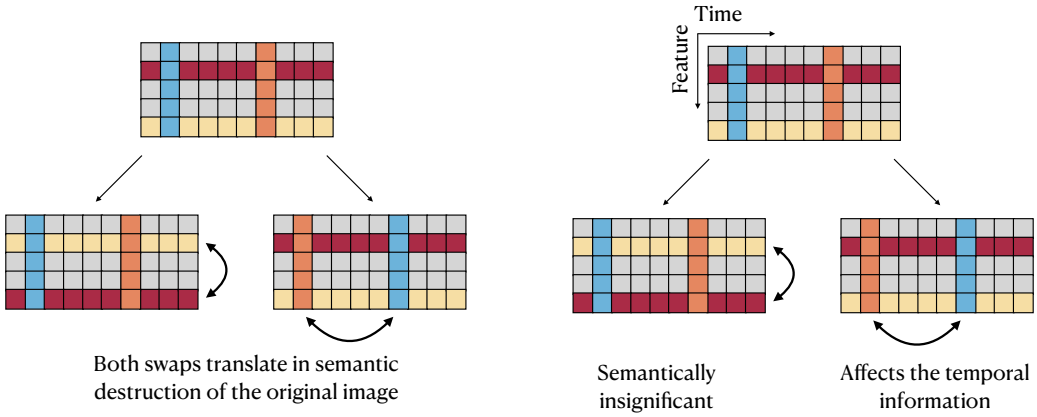


Figure 6.1: Inconsistent saliency map explanation in time series predictions. Orange cells represent important timestamps and features for predicting certain sliding windows. The red-marked periods are overlapped across the three adjacent sliding windows with the same input time series values. However, their feature importance is inconsistent.

explanation methods [IGCBF20, PHC21b], the quality of such explanations is often questionable. In this chapter, we address two main issues that arise when using saliency maps for explaining time series data predictions, i.e., the **consistency** and the **robustness**, and show in the experiments that several common feature attribution approaches are neither consistent nor robust.

Deep time series prediction models usually consider sliding windows as basic input units to capture temporal information. Analog to saliency maps for visualizing image pixel importance, similar saliency maps are generated for time series frames with *feature-time pixel* importance. Current research on explanation approaches for time series data can be classified into two categories. The first category contains methods treating sliding windows as *frames* of images and applying classical image explanation methods, e.g., SHAP [LL17], LIME [RSG16], and DeepLIFT [SGK17]. Those methods extract local feature information while neglecting the time series data's typical time structure. The second category contains methods considering the time dimension as an additional feature for separate explanations [BSC+21, IGCBF20]. Overall, explanation methods on time series data consider the time dimension either jointly with the other features [PHC21b] or separately through sequentially considering time and features [IGCBF20]. In both cases so far, the explanations consider one single *frame* (i.e., sliding window). Hence, both categories reveal insufficient interpretation of the overall temporal infor-



(a) Image data **(b)** Time series

Figure 6.2: Non-robust saliency maps in time series predictions. The position of a pixel in an image is defined by row and column numbers, while in time series data, by input variables and timestamps. Swaps of rows and columns in images may affect the semantic meaning of the entire data frame. For time series, this may happen only when swapping observations at different timestamps.

mation over a long time span. We claim that explanations over the intersection of sliding windows should exhibit *consistent* behaviors to identify such a flaw in current time series saliency explanations. We admit that in adjacent sliding windows, different temporal contexts may lead to different absolute feature attribution. Therefore, we pursue the consistent *relative attribution* in local sub-windows. Figure 6.1 illustrates the *inconsistency* effect among overlapping time windows.

In addition to the saliency explanation consistency, the *robustness* of saliency maps against feature perturbation is another essential factor in ensuring explanation quality. As shown in Figure 6.2a, the semantic meanings of columns and rows are equivalent in image data. At the same time, in time series, the time structure makes time series data semantically different and introduces dependence among the observations in the various timestamps. In images, swaps of both rows and columns affect the semantic structure of the original data. In contrast, in time series, only the swaps affecting the temporal orders of the observations are semantically significant. The order in which the input features are organized has no semantic effect. The phenomenon is illustrated in Figure 6.2b, where the x-axis corresponds to time and the y-axis to the input variables. When saliency maps are applied to time series, the salient features should be insensitive to the order of input features. We call this the *robustness* of saliency explanations.

We study the *consistency* and *robustness* of saliency explanations for time series predictions. We examine saliency explanations from popularly used perturbation- and gradient-based approaches [SHJ+17, STY17] on multiple deep classification models [HS97, LFV+17, VSP+17]. We show on five real-world datasets that the studied saliency explanations suffer from consistency and robustness issues. These preliminary results underline the problems encountered as a motivation to conduct further research on time series prediction explanation.

6.2 Related works

Intrinsic and post-hoc explanations have transformed into a core topic for machine learning research. The interest in increasing the explainability of the methods embraces the entire machine learning and AI communities [SGK17, SK10]. GRAD [BSH+10] is a post-hoc agnostic interpretation method that is able to explain nonlinear classifiers at a local level. The explanations measure how each data point has to be moved to change the predicted label [BSH+10]. The local scores are derived from the direct computation of the local gradients (or their estimations). Similarly, Integrated Gradients [STY17] is also a gradient-based feature importance attribution method and builds up on [BSH+10] and two axioms, i.e., the *sensitivity* and *implementation invariance*. Finally, SHAP [LL17] represents a successful attempt to introduce Shapley values in machine learning. Lundberg et al. [LL17] use Shapley values to assign importance scores to features for local explanations of black-box models' predictions. The Shapley values' approximations are based on the computations of the gradient of model predictions.

Although progress is not neglectable, the explanations provided by the most recent works are mostly not quantifiable, thus still raising trust issues [ZSS+19]. Few recent works focus on the quality of the explanation methods; Dombrowski et al. [DAA+19] show that explanations for image classification are non-robust against possible visually hardly detectable manipulations.

Explanation methods appear for most machine learning techniques with different strengths. Time series represents a data type where most implemented methods still lack explanations. One of the reasons for the poor literature on the explainability of time series data is the additional time-dependent structure. Explanations often reduce to applications of model-agnostic post-hoc explanations for general data samples to time-dependent data. The time structure is often disregarded, and the timestamps are treated as independent samples on which the model is learned [BSC+21, SAEA+19]. Another thread of approaches using attention-based models

obtains time-dependent explanations by attention weights [KZK+19, SRTS18, CBS+16]. The acquired feature and time attribution to the prediction can be visualized in saliency maps, which are initially implemented for images [BBM+15] and are a current trend in obtaining explanations for importance scores of timestamps and features. For example, gradient-based [BSH+10, STY17, STK+17, SGK17] and perturbation-based feature importance scores [ZF14, SHJ+17]. Ismail et al. [IGCBF20] pointed out how these methods often suffer from a lack of understanding of the time-feature structures, either allowing them to achieve only good performances at a time level or the features level. The authors propose an alternative two-step approach to saliency explanations for time series, where the time structure is first considered, and the importance of the features is considered in the second step. The explanation quality of such a method is still under-studied in the time series domain.

6.3 Open issues on saliency explanations

In this section, we formally define the consistency and robustness of saliency explanations for time series predictions.

We indicate with $X = (X_1, \ldots, X_N)$ a multivariate N-dimensional discrete time series where X_i is the i-th univariate dimension; t_0 is the first timestamp on which the time series is defined. For each timestamp $t_k > t_0$, $X(t_k)$ is a N-dimensional vector of real values, i.e., $X(t_k) \in \mathbb{R}^N$. We study the consistency and robustness of saliency explanations for classification models trained on time series data. We draw upon the concept of *consistency* proposed by Pillai et al. [PKO+22], and define *consistency* of saliency explanations over adjacent sliding windows of time series. Additionally, regarding *robustness*, we consider the influence of swaps of *features* in the series.

6.3.1 Consistency

We define *time windows* $\{w_s^d\}_{s\in\mathbb{N}}$ dependent on the window length $d \in \mathbb{N}$ and the starting timestamp t_s, i.e.,

$$w_s^d = \{t_s, \ldots, t_{s+d-1}\} \tag{6.1}$$

For each time window and given a fixed saliency map method assigning importance score S, we get $S(w_s^d) = S_s^d$ a matrix in $\mathbb{R}^{N\times d}$ such that $(S_s^d)_{n,t}$ is the importance scores assigned to the input variable X_n at time t. Saliency maps are transposed from image (pre)processing applications to explain time series classification predictions. We examine the consistency of

saliency maps defined over overlapping windows. Given two windows w_s^d and $w_{\bar{s}}^{\bar{d}}$ such that $|w_s^d \cap w_{\bar{s}}^{\bar{d}}| \neq \emptyset$ and the respective saliency maps S_s^d and $s_{\bar{s}}^{\bar{d}}$, the saliency explanations are inconsistent at timestamp t, if $t, \bar{t} \in w_s^d \cap w_{\bar{s}}^{\bar{d}}$ such that

$$(S_s^d)_{n,t} > (S_{\bar{s}}^{\bar{d}})_{n,t} \text{ and } (S_s^d)_{n,\bar{t}} < (S_{\bar{s}}^{\bar{d}})_{n,\bar{t}}, \tag{6.2}$$

i.e., the importance scores assigned to features and timestamps are *relatively* inconsistent among overlapping time windows.

6.3.2 Robustness

Although similarly structured, images and time series intrinsically contain different semantic meanings due to the time dependency. However, the time series explanation should be insensitive to the feature ordering. A saliency explanation is considered as *robust* if the saliency changes accordingly when the features are swapped. We define the feature swapping operation on data window w_s^d and observe the effect in the corresponding saliency explanation S_s^d. Concretely, we swap a random pair of features X_i and X_j ($i \neq j$) in w_s^d for all timestamps from t_s to t_{s+d-1}. Their feature attribution in S_s^d are $(S_s^d)_i$ and $(S_s^d)_j$. After features swapping, the data window is denoted by w^{*d}_s, and the new saliency explanation is S^{*d}_s. $(S_s^d)_i$ corresponds to $(S^{*d}_s)_j$ while $(S_s^d)_j$ corresponds to $(S^{*d}_s)_i$. The saliency explanations are robust if $t_1, t_2 \in w_s^d \cap w_s^d$ such that

$$(S_s^d)_{i,t_1} > (S_s^d)_{i,t_2} \text{ and } (S^{*d}_s)_{j,t_1} > (S^{*d}_s)_{j,t_2}, \tag{6.3}$$

i.e., important feature-time pixels maintain relative importance after swapping the feature of the data window.

6.4 Experiments

We perform experiments on real-world datasets for time series classification. We generate various types of explanations in the form of saliency maps for the predictions made by the classifiers to examine their consistency and robustness. We incorporate artificial padding into the input sequences to precisely control the feature importance and simulate the sliding window mechanism commonly used in time series prediction tasks. This section presents our findings on identifying inconsistency and non-robust saliency explanations across multiple datasets.

6.4.1 Experiment setup

6.4.1.1 Dataset description

We consider five real-world univariate time series datasets: *Power Demand* (PD), *Wine* (WIN), *Italy Power Demand* (IPD), *Two Lead ECG* (ECG) and *Mote Strain* (MS). PD derives from Keogh et al. [KLF05], while the others are available in the UCR Archive [CKH+15]. We preprocess all datasets by dividing them into non-overlapping windows a priori, and the class labels of each window are available. However, the ground truth does not include the attribution of the prediction. In Sections 6.4.2.1 and 6.4.2.2, we introduce artificial padding with random noise to each input window and assign equal importance to the area of the original input.

6.4.1.2 Experiment configuration

For our experiments, we select three representatives from the common saliency explanation approaches [IGCBF20] for time series data. We employ *Feature Permutation* (FP) and *Feature Ablation*(FA) [SHJ+17], which are perturbation-based methods, and *Integrated Gardients* (IG) [STY17], which is a gradient-based method. We use the implementation provided by Ismail et al. [IGCBF20].

We investigate the behavior of saliency explanations on three types of network structures: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and attention-based networks. To this end, we pick three implementations commonly used for time series data: LSTM [HS97], TCN [LFV+17], and Transformer [VSP+17]. We configure these models with a Softmax output layer for classification and train the models on all the padded variants of the input windows, including top, middle, and bottom padding. During the test phase, we generate saliency maps and analyze the effect of each group of padding variants separately.

6.4.2 Performance

6.4.2.1 Consistency evaluation

We apply artificial padding to each univariate time window to evaluate the explanation consistency over sliding windows. Specifically, we expand each univariate data window $w_s^d \in \mathbb{R}^{n \times d}$ to a matrix $m \in \mathbb{R}^{\alpha \times \lfloor \beta \cdot d \rfloor} (\alpha > n, 1 < \beta < 3)$. The data window w_s^d is placed on d consecutive dimensions of m, and the other dimensions are filled with randomly sampled noise from a normal distribution. The effect of a sliding window can be simulated by placing w_s^d at different

rows in m. Specifically, we allocate w_s^d at the top, middle, and bottom third of m to generate three overlapping sliding windows, i.e., three variants of each input window. We call the area in the saliency map corresponding to the input window w_s^d, the *area of interest*. We show the experimental results by setting $\alpha = 4$ and $\beta = \frac{5}{3}$. An example of the padded data window is shown in Figure 6.3.

In the proposed construction, each padding variant group (top/middle/bottom) contains the same input window but is only located differently. To examine the consistency of the saliency explanations, we compare the feature ranking of the obtained attributions at corresponding locations in each padding variant. As a showcase, we visualize the result of one window from the IPD dataset in Figure 6.3. The upper left group of saliency maps in Figure 6.3 represent the input window at various padding positions, the y-axis being the time and the x-axis being the input features. Only the second feature contains essential information to be learned by the classifiers. The remaining groups of saliency maps correspond to the three explanation models FP, FA, and IG. We expect the second feature column's top, middle, and bottom third to be marked as salient. However, as Ismail et al. [IGCBF20] have already shown, classical saliency methods might fail on time series data due to the temporal feature, and our experiment results in Figure 6.3 confirm their claim where the latest timestamps play more important roles in the prediction, especially for LSTM. The explainers suffer from distinguishing important features in TCN and Transformers.

Despite the sub-optimal saliency explanations, we analyze the consistency between the padding variants. We empirically evaluated the disagreement on the saliency explanations using Kendall's τ [Ken48] and Pearson correlation. Kendall's τ measures the smallest number of swaps of adjacent elements that transform one ranking into the other, while the Pearson correlation coefficient measures the covariance of the two random variables normalized by the product of their standard deviations. All quantities can be estimated using finite samples.

We calculate the importance scores for each timestamp and input feature, obtaining the importance ordering of the *area of interest*. For each pair of ranking from the three padding variants, we analyze the pairwise comparisons among rankings of feature-time pixels in the saliency explanations of FP, FA, and IG. The average Kendall's τ and Pearson correlation (ρ) are summarized in Table 6.1 and the absolute values are visualized in Figure 6.4.

Table 6.1 contains, for each data set, neural network architecture, and saliency map, the average Kendall's τ and Pearson correlation coefficients with the respective variance. From the table, it is easy to spot how the importance scores rankings vary in ranges below 1. Kendall's

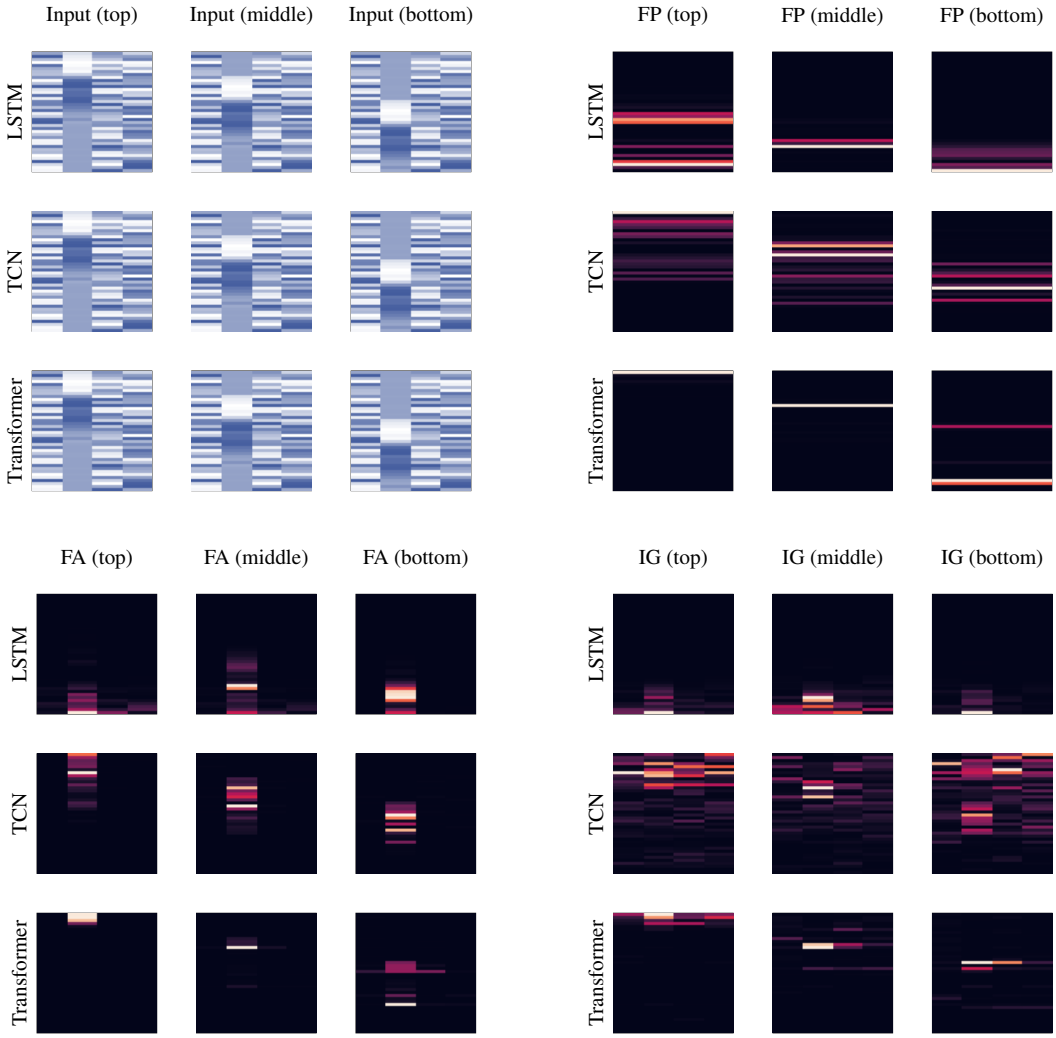
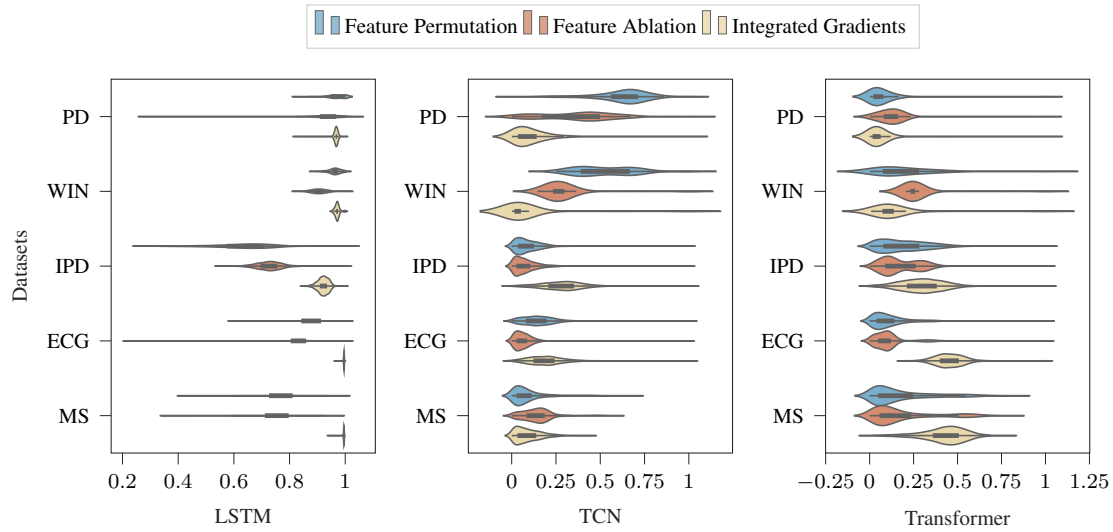
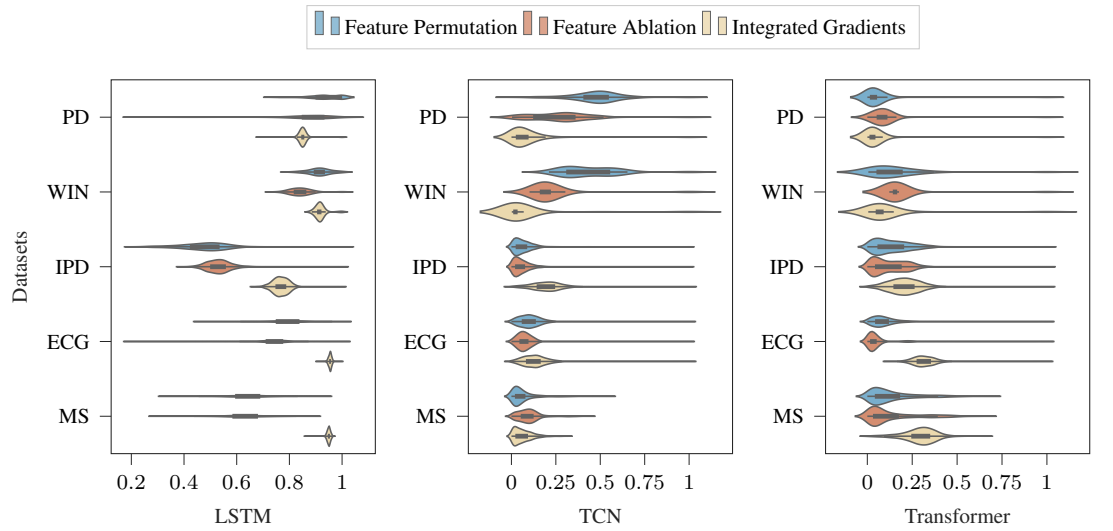


Figure 6.3: Saliency explanation of one data window from the IPD dataset: the upper left group blue heatmaps denote the variants of one input frame from the IPD dataset, where rows are timestamps and columns are features. The real data window is located at the frame's top, middle, and bottom third of the second feature, and the rest of the elements are random noise. The other three groups of saliency maps are acquired from three feature attribution algorithms: Feature Permutation (FP), Feature Ablation (FA), and Integrated Gradient (IG).



(a) Violin plots of Pearson correlation absolute value



(b) Violin plots of Kendall's tau absolute value

Figure 6.4: Violin plots of inconsistent saliency explanations.

τ and Pearson correlation coefficients range between 1 and -1, where 1 indicates complete agreement among the rankings, while values close to zero suggest non-constant and independent orderings. From Figures 6.4a and 6.4b, we observe that the coefficients are, in most cases, crowded at low values, and only rare cases show good agreement among the obtained rankings in the different windows.

A special case is the LSTM model, which provides consistent saliency maps among the various windows. However, observing the explanations of the LSTM model, we see that both Kendall's τ and Pearson correlation coefficients tend to accumulate to high scores (≈ 1) as the LSTM model tends to accumulate the learning in the last timestamps, thus implying that the explanation methods assign high importance only to the last timestamps. We observed that FA correctly finds the relevant timestamps but cannot distinguish between noisy and relevant features.

In addition to the relative ranking, we also check the quality of saliency explanations using Recall@k. Table 6.2 contains the Recall@k obtained among the importance rankings of timestamps in the *areas of interest*. Recall@k measures the ratio among correctly relevant and retrieved elements and the number of relevant elements and ranges in $[0, 1]$. High recall (≈ 1) indicates that the highly ranked feature-time pixels are concentrated in the *area of interest*, while low Recall@k indicates the inability to find relevant elements correctly.

6.4.2.2 Robustness evaluation

To evaluate the robustness of saliency explanations, we apply the feature swapping depicted in Figure 6.2. Specifically, we continue using the padded input matrix $m \in \mathbb{R}^{\alpha \times \lfloor \beta \cdot d \rfloor}$ from Section 6.4.2.1 and swap the feature dimensions containing the original input data window (*area of interest*) with noise dimensions. We train different classification models with the swapped and not swapped data. For simplicity, we always locate the original window in the middle of the selected feature dimension in this experiment. We compare the ranking of feature-time pixel explanations in the *areas of interest* of swapped and not swapped pairs. The Kendall's τ and Pearson correlation ρ are summarized in Table 6.3.

The absolute values of Kendall's τ and Pearson correlation for TCN and Transformer models indicate a significant difference in saliency maps after the swapping. In other words, when the important feature is switched with a noisy feature, the feature attribution in the saliency map is not switched correspondingly. An exception is the LSTM model, which robustly explains all datasets except IPD. However, the explanation quality is limited.

6.5 Conclusion

While explanations based on saliency maps have succeeded in vision and natural language domains, they remain challenging for time series data. In addition to the well-known challenges

posed by the additional time dimension to the input features, in this chapter, we have also identified issues on *consistency* and *robustness*.

We explored the issue of *inconsistency* raising in saliency explanations over overlapping time windows and the issue of *non-robustness* when swapping features in time series windows. The presented exploratory analysis aims to raise awareness of the described problems and motivates further development of saliency methods that address the existing flaws.

Table 6.1: Consistency ranking analysis

		Feature Permutation (FP)		Feature Ablation (FA)		Integrated Gradients (IG)	
		τ	ρ	τ	ρ	τ	ρ
PD	LSTM	$0.936^{\pm 0.063}$	$0.967^{\pm 0.035}$	$0.868^{\pm 0.114}$	$0.921^{\pm 0.092}$	$0.852^{\pm 0.024}$	$0.968^{\pm 0.012}$
	TCN	$0.475^{\pm 0.143}$	$0.625^{\pm 0.155}$	$0.247^{\pm 0.200}$	$0.334^{\pm 0.256}$	$0.060^{\pm 0.148}$	$0.080^{\pm 0.168}$
	Transformer	$0.040^{\pm 0.136}$	$0.049^{\pm 0.145}$	$-0.049^{\pm 0.149}$	$-0.086^{\pm 0.161}$	$0.026^{\pm 0.138}$	$0.030^{\pm 0.145}$
WIN	LSTM	$0.914^{\pm 0.042}$	$0.961^{\pm 0.023}$	$0.843^{\pm 0.044}$	$0.908^{\pm 0.030}$	$0.918^{\pm 0.022}$	$0.971^{\pm 0.009}$
	TCN	$0.445^{\pm 0.164}$	$0.530^{\pm 0.169}$	$0.220^{\pm 0.159}$	$0.287^{\pm 0.149}$	$0.052^{\pm 0.200}$	$0.057^{\pm 0.206}$
	Transformer	$0.150^{\pm 0.192}$	$0.196^{\pm 0.207}$	$-0.11^{\pm 0.218}$	$-0.198^{\pm 0.236}$	$0.107^{\pm 0.180}$	$0.140^{\pm 0.178}$
IPD	LSTM	$0.475^{\pm 0.086}$	$0.633^{\pm 0.101}$	$0.531^{\pm 0.047}$	$0.725^{\pm 0.045}$	$0.767^{\pm 0.030}$	$0.921^{\pm 0.019}$
	TCN	$0.001^{\pm 0.083}$	$-0.002^{\pm 0.112}$	$-0.004^{\pm 0.075}$	$-0.031^{\pm 0.095}$	$0.192^{\pm 0.083}$	$0.277^{\pm 0.114}$
	Transformer	$0.124^{\pm 0.115}$	$0.167^{\pm 0.157}$	$0.054^{\pm 0.141}$	$0.042^{\pm 0.199}$	$0.204^{\pm 0.088}$	$0.295^{\pm 0.121}$
ECG	LSTM	$0.789^{\pm 0.070}$	$0.873^{\pm 0.056}$	$0.738^{\pm 0.062}$	$0.827^{\pm 0.055}$	$0.954^{\pm 0.007}$	$0.995^{\pm 0.001}$
	TCN	$0.102^{\pm 0.074}$	$0.143^{\pm 0.096}$	$0.072^{\pm 0.062}$	$0.054^{\pm 0.070}$	$0.129^{\pm 0.078}$	$0.189^{\pm 0.101}$
	Transformer	$0.089^{\pm 0.082}$	$0.098^{\pm 0.110}$	$0.020^{\pm 0.092}$	$-0.038^{\pm 0.137}$	$0.315^{\pm 0.066}$	$0.453^{\pm 0.081}$
MS	LSTM	$0.642^{\pm 0.072}$	$0.768^{\pm 0.066}$	$0.632^{\pm 0.078}$	$0.753^{\pm 0.070}$	$0.950^{\pm 0.007}$	$0.995^{\pm 0.002}$
	TCN	$0.038^{\pm 0.096}$	$0.053^{\pm 0.134}$	$-0.031^{\pm 0.116}$	$-0.058^{\pm 0.167}$	$0.055^{\pm 0.061}$	$0.081^{\pm 0.089}$
	Transformer	$0.124^{\pm 0.136}$	$0.156^{\pm 0.189}$	$0.093^{\pm 0.165}$	$0.076^{\pm 0.244}$	$0.291^{\pm 0.089}$	$0.423^{\pm 0.125}$

Table 6.2: Consistency Recall@k

		Feature Permutation (FP)			Feature Ablation (FA)			Integrated Gradients (IG)		
		Top	Middle	Bottom	Top	Middle	Bottom	Top	Middle	Bottom
PD	LSTM	0.041	0.000	0.000	0.072	0.000	0.000	0.165	0.268	0.268
	TCN	0.227	0.216	0.216	0.454	0.320	0.320	0.103	0.206	0.206
	Transformer	0.258	0.258	0.258	0.825	0.928	0.928	0.351	0.330	0.330
WIN	LSTM	0.064	0.051	0.051	0.103	0.060	0.060	0.244	0.248	0.248
	TCN	0.256	0.269	0.269	0.500	0.487	0.487	0.286	0.295	0.295
	Transformer	0.333	0.812	0.812	0.949	0.962	0.962	0.389	0.385	0.385
IPD	LSTM	0.250	0.250	0.250	0.625	0.708	0.708	0.375	0.458	0.458
	TCN	0.250	0.250	0.250	0.875	0.958	0.958	0.292	0.375	0.375
	Transformer	0.250	0.250	0.250	0.750	0.708	0.708	0.167	0.292	0.292
ECG	LSTM	0.171	0.171	0.171	0.341	0.244	0.244	0.256	0.268	0.268
	TCN	0.256	0.256	0.256	0.634	0.634	0.634	0.329	0.305	0.305
	Transformer	0.256	0.256	0.256	0.780	0.829	0.829	0.244	0.293	0.293
MS	LSTM	0.250	0.262	0.262	0.536	0.560	0.560	0.357	0.321	0.321
	TCN	0.250	0.262	0.262	0.750	0.798	0.798	0.298	0.381	0.381
	Transformer	0.250	0.250	0.250	0.845	0.821	0.821	0.274	0.369	0.369

Table 6.3: Robustness ranking analysis

		Feature Permutation (FP)		Feature Ablation (FA)		Integrated Gradients (IG)	
		τ	ρ	τ	ρ	τ	ρ
PD	LSTM	$0.972^{\pm 0.064}$	$0.985^{\pm 0.036}$	$0.973^{\pm 0.112}$	$0.982^{\pm 0.102}$	$0.807^{\pm 0.064}$	$0.942^{\pm 0.066}$
	TCN	$0.680^{\pm 0.168}$	$0.803^{\pm 0.157}$	$0.597^{\pm 0.259}$	$0.706^{\pm 0.285}$	$0.221^{\pm 0.148}$	$0.307^{\pm 0.175}$
	Transformer	$0.042^{\pm 0.141}$	$0.052^{\pm 0.153}$	$0.129^{\pm 0.148}$	$0.164^{\pm 0.161}$	$0.123^{\pm 0.133}$	$0.171^{\pm 0.143}$
WIN	LSTM	$0.932^{\pm 0.047}$	$0.971^{\pm 0.021}$	$0.924^{\pm 0.056}$	$0.967^{\pm 0.029}$	$0.929^{\pm 0.075}$	$0.972^{\pm 0.057}$
	TCN	$0.598^{\pm 0.107}$	$0.696^{\pm 0.083}$	$0.553^{\pm 0.094}$	$0.676^{\pm 0.073}$	$0.083^{\pm 0.186}$	$0.105^{\pm 0.189}$
	Transformer	$0.267^{\pm 0.241}$	$0.347^{\pm 0.283}$	$0.466^{\pm 0.153}$	$0.587^{\pm 0.167}$	$0.345^{\pm 0.149}$	$0.473^{\pm 0.143}$
IPD	LSTM	$0.446^{\pm 0.124}$	$0.603^{\pm 0.147}$	$0.579^{\pm 0.075}$	$0.732^{\pm 0.076}$	$0.699^{\pm 0.062}$	$0.864^{\pm 0.053}$
	TCN	$0.114^{\pm 0.141}$	$0.157^{\pm 0.188}$	$0.277^{\pm 0.159}$	$0.358^{\pm 0.194}$	$0.254^{\pm 0.137}$	$0.361^{\pm 0.183}$
	Transformer	$0.369^{\pm 0.158}$	$0.480^{\pm 0.178}$	$0.364^{\pm 0.215}$	$0.458^{\pm 0.244}$	$0.322^{\pm 0.148}$	$0.449^{\pm 0.192}$
ECG	LSTM	$0.821^{\pm 0.100}$	$0.898^{\pm 0.073}$	$0.841^{\pm 0.133}$	$0.905^{\pm 0.103}$	$0.967^{\pm 0.029}$	$0.996^{\pm 0.028}$
	TCN	$0.382^{\pm 0.119}$	$0.515^{\pm 0.138}$	$0.517^{\pm 0.087}$	$0.657^{\pm 0.101}$	$0.207^{\pm 0.125}$	$0.300^{\pm 0.172}$
	Transformer	$0.276^{\pm 0.126}$	$0.365^{\pm 0.153}$	$0.535^{\pm 0.141}$	$0.673^{\pm 0.141}$	$0.415^{\pm 0.081}$	$0.582^{\pm 0.101}$
MS	LSTM	$0.747^{\pm 0.112}$	$0.857^{\pm 0.089}$	$0.694^{\pm 0.107}$	$0.811^{\pm 0.090}$	$0.968^{\pm 0.027}$	$0.996^{\pm 0.024}$
	TCN	$0.147^{\pm 0.140}$	$0.201^{\pm 0.187}$	$0.184^{\pm 0.144}$	$0.248^{\pm 0.189}$	$0.068^{\pm 0.087}$	$0.100^{\pm 0.126}$
	Transformer	$0.473^{\pm 0.220}$	$0.575^{\pm 0.238}$	$0.513^{\pm 0.215}$	$0.627^{\pm 0.239}$	$0.453^{\pm 0.105}$	$0.621^{\pm 0.125}$

Table 6.4: Robustness Recall@k

		Feature Permutation (FP)			Feature Ablation (FA)			Integrated Gradients (IG)		
		Top	Middle	Bottom	Top	Middle	Bottom	Top	Middle	Bottom
PD	LSTM	0.031	0.000	0.000	0.072	0.000	0.000	0.134	0.237	0.237
	TCN	0.227	0.258	0.258	0.299	0.443	0.443	0.155	0.216	0.216
	Transformer	0.268	0.309	0.309	0.742	0.876	0.876	0.268	0.299	0.299
WIN	LSTM	0.056	0.030	0.030	0.103	0.051	0.051	0.231	0.248	0.248
	TCN	0.274	0.265	0.265	0.389	0.355	0.355	0.295	0.282	0.282
	Transformer	0.303	0.252	0.252	0.466	0.568	0.568	0.179	0.372	0.372
IPD	LSTM	0.250	0.250	0.250	0.458	0.417	0.417	0.375	0.458	0.458
	TCN	0.250	0.250	0.250	0.458	0.500	0.500	0.333	0.375	0.375
	Transformer	0.250	0.250	0.250	0.458	0.542	0.542	0.292	0.333	0.333
ECG	LSTM	0.244	0.134	0.134	0.341	0.232	0.232	0.256	0.280	0.280
	TCN	0.256	0.256	0.256	0.476	0.476	0.476	0.232	0.256	0.256
	Transformer	0.256	0.256	0.256	0.634	0.622	0.622	0.390	0.305	0.305
MS	LSTM	0.250	0.250	0.250	0.381	0.417	0.417	0.333	0.262	0.262
	TCN	0.250	0.250	0.250	0.500	0.500	0.500	0.405	0.381	0.381
	Transformer	0.250	0.250	0.250	0.357	0.405	0.405	0.190	0.226	0.226

7

Example-based Explanation

In this chapter, we showcase a concrete time series anomaly interpretation approach ProtoAD. ProtoAD learns prototypes end-to-end during model training and employs them as example-based explanations for the normal patterns during anomaly detection. Without significant impact on the detection performance, prototypes shed light on the deep black-box models and provide intuitive understanding for domain experts and stakeholders. We extend the widely used prototype learning in classification problems into anomaly detection. By visualizing both the latent space and input space prototypes, we intuitively demonstrate how normal data are modeled and why specific patterns are considered abnormal. As an example-based explanation approach, ProtoAD provides an intuitive human-understandable interpretation of the time series anomalies and, hence, is valuable in real-world applications.

7.1 Introduction

As introduced in Chapter 4, Autoencoders are widely used for unsupervised anomaly detection tasks. Despite the performance advantage, Autoencoders also suffer from criticisms of other deep neural networks, such as the lack of transparency. Though the detected anomalies

fit well with standard evaluation criteria, their reliability is not well-proven. The lack of human interpretable information makes it hard to tell why an anomaly is abnormal, especially in high-dimensional time series data or long input sequences. To address the black-box issue, we use example-based prototypes as an intuitive and explainable solution for interpreting anomalies in time series data. Prototypes are widely used for case-based reasoning in computer vision [LLCR18, CLT+18, HCLR19], graph learning [ZLW+21] and sequential data learning [NCC+21, MXQR19, GGOGP19]. Different from Chapter 5, where we learn prototypes using the K-Medoid algorithm as a prior step, here we learn prototypes in an end-to-end fashion by embedding a prototype layer in the Autoencoder. With the built-in prototype layer in the neural network, prototype-based models are efficient in training without requiring extra investigation into the interpretability functionalities. Moreover, the prototypes are usually self-contained and straightforward, e.g., representative animal faces, sentences, and sensor data patterns. However, prototypes are still understudied in the anomaly detection field.

We propose using prototypes to interpret normal data during anomaly detection using Autoencoders. In this context, data showing certain repeating normal patterns is generated by one or more latent distributions, while the anomaly is any data point or period that deviates from these normal patterns. We model the normal patterns of the time series data with prototypes in the latent space of the Autoencoder and learn multiple prototypes to discover the latent components of the normal data distribution. Anomaly patterns that lie distantly from the normal patterns in the latent space can be explained by comparing them with the learned prototypes.

The main contribution of this chapter can be summarized as follows:

1. we propose ProtoAD, an end-to-end LSTM-Autoencoder for anomaly detection with prototype learning

2. we develop latent space prototype-based explanations for the understanding of the normal state of the studied data

3. we evaluate our method with synthetic and real-world time series data. Moreover, we visually demonstrate prototype examples to qualitatively show our model's benefit.

7.2 Related works

7.2.1 Reconstruction-based anomaly detection

Autoencoders have been used as an unsupervised anomaly detection approach for years. Feed-forward Autoencoders [ZSM+18] and Variational Autoencoders [XCZ+18] are used for time-independent data. In contrast, RNN-based Autoencoders [MRA+16] show their strength in detecting contextual anomalies in time series data. Based on the reconstruction error, a standard approach for estimating anomaly likelihood is to assume the reconstruction error following a normal distribution and measure the Mahalanobis distance between the reconstruction error of unknown data and the estimated distribution [MRA+16]. In addition to reconstruction error, the hidden representation in the latent space can also be used for likelihood estimation [ZSM+18]. Gaussian Mixture Model (GMM) [ZSM+18] and energy-based model [ZCLZ16] have also been used for the likelihood estimation. Common thresholding techniques over the anomaly likelihood are based on maximizing the performance on a validation set, which requires labels in advance [MRA+16].

Other approaches, including the hierarchical temporal memory (HTM) [ALPA17] and temporal convolutional network (TCN) [HZ19] have also been adopted in TSAD concerning different use cases and data properties. However, they are not directly relevant to the reconstruction-based models.

7.2.2 Explanation with prototypes

Due to the complex properties of both feature and time dimensions of time series data, prototypes are considered an intuitive explanation. Common prototype learning approaches for neural networks follow a three-step paradigm: 1) representation learning, 2) prototype learning in the latent space, and 3) class prediction. The objective commonly includes 1) minimizing classification error, 2) minimizing the distances between each hidden representation and one of the prototypes, and 3) maximizing the distances between prototypes.

In the existing prototype learning literature, Li et al. [LLCR18] employ a multi-layer convolutional neural network to construct the Autoencoder, which learns hidden representations for image data. They rely on the decoder to project the learned prototypes in the human-understandable space, though sometimes producing unrealistic reconstructions. Using a single encoder to replace the Autoencoder is considered as a reduction of training effort

in [CLT+18, MXQR19], and the authors use the nearest neighborhood of each prototype in the latent space as the corresponding realistic patterns in the input space. Chen et al. [CLT+18] and Hase et al. [HCLR19] build up the encoder with convolutional neural networks to encode image data, Ming et al. [MXQR19] use RNNs for sequential data, Ni et al. [NCC+21] use a convolutional layer to learn time series representations, Zhang et al. [ZLW+21] employ graph neural networks for the encoder. In our work, we use the LSTM-Autoencoder for both reconstruction-based TSAD and hidden space representation learning.

The standard objective functions of existing prototype learning approaches consist of multiple regularisation terms that are trained jointly. To ensure the representation ability of the prototypes, many existing works [LLCR18, MXQR19, GGOGP19, ZLW+21, CLT+18] minimize the distance between each prototype and every nearby hidden representation as well as every hidden representation to each prototype. Furthermore, the learned prototypes are supposed to be diverse from each other [MXQR19, ZLW+21, GGOGP19]. In the objective function, we follow the standard design of the regularization terms above. However, different from most existing works, which use cross-entropy for their classification tasks to minimize the classification error [LLCR18, MXQR19, HCLR19], in our unsupervised setting, we use the reconstruction-error to regularize the reconstruction process of normal data.

Besides prototypes, other techniques have also been used to explain time series data. The representative subsequences Shapelets [KML20, LCX+20] can be similarly used for explanation. Instead of finding the representative patterns as prototypes, counterfactuals [AALC21] explain the instance towards the opposite class.

7.3 Methodology

7.3.1 Preliminaries

7.3.1.1 Terminology

Let $X = \{X_t\}_{t \in \mathbb{Z}}$ be a d-dimensional time series that shows a regularly repeating pattern over time periods of some length L. These repeating patterns are contained in sliding windows $W_t = \{X_{t+1}, \ldots, X_{t+L}\}$ of L consecutive elements of the time series. Often, the window size L can be selected based on prior knowledge of the dataset, which is known to show seasonality over, for example, one day or one week.

In ProtoAD, we consider both point and contextual anomalies. We assume that the data points are generated by a periodically stationary time series X with periodicity L [WSM15, UD09]. The time series consists of regularly repeating patterns of length L, which evolve over time without distributional changes, i.e., we do not consider concept drifts [GŽB+14].

Let $(W_t, y_t)_{t\in\mathbb{Z}}$ be the dataset after applying the sliding window and $y_i \in \{0, 1\}$ is the label of the window W_t (0 for normal data and 1 for anomaly). The anomaly detection is conducted on the window level. A window is considered abnormal if at least one point or a sub-window with multiple points shows significantly different behavior from the normal windows. The significance is determined by comparing the window anomaly score predicted by the model and a user-defined threshold.

7.3.1.2 Problem statement

Given the multi-dimensional time series data with applied sliding windows, the target is to train an Autoencoder-based end-to-end anomaly detector that

1. detect abnormal windows in an unsupervised manner
2. learn representative prototypes of normal data in the latent space
3. leverage interpretation of anomalies based on the prototypes of normal data.

7.3.2 Architecture overview

In ProtoAD, we use an LSTM-Autoencoder to learn time series hidden representations in the latent space and feed the representations to the prototype layer for similarity-based prototype comparison. Specifically, we design the architecture and training procedure for unsupervised anomaly detection, while only normal data is used for the training and prototype learning. An overview of the ProtoAD architecture is shown in Figure 7.1.

We construct the LSTM-Autoencoder in the fashion of [MRA+16]. More specifically, the d dimensional input window $W_t = \{X_{t+1}^{t+L}\}$ is feed into the encoder

$$\mathbf{f}\colon \mathbb{R}^{L\times d} \to \mathbb{R}^m \tag{7.1}$$

The last hidden state of the encoder LSTM unit $h_i = \mathbf{f}(W_t)$ ($h_i \in \mathbb{R}^m$) is used as the hidden representation of the input window in latent space. A same-structured decoder

$$\mathbf{g}\colon \mathbb{R}^m \to \mathbb{R}^{L\times d} \tag{7.2}$$

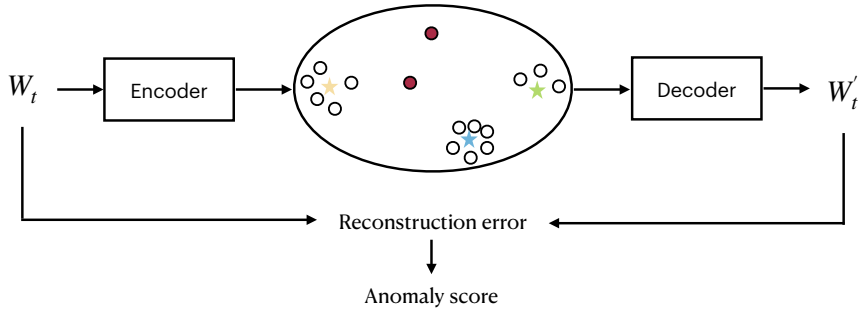


Figure 7.1: ProtoAD overview: ProtoAD consists of an encoder and a decoder, which reconstructs the input data window W_t to $W_t^{'}$. The anomaly score is calculated based on the reconstruction error. In the latent space, prototypes (stars) are learned as representative points of the encoded normal windows (white dots). In the test phase, latent representations of the abnormal data windows (red dots) can also be explained by comparing them with the learned prototypes.

targets at reconstructing the window $W_t^{'} = \{X_{t+1}^{'t+L}\}$ from the hidden representation. The decoder LSTM unit takes h_i as the initial hidden state while taking the real data from the previous timestamp as input. We train the Autoencoder to minimize the reconstruction error of normal windows, i.e., no anomaly data will be used during training.

The reconstruction error at timestamp t is defined as

$$e_t = |X_t - X_t^{'}| \tag{7.3}$$

The train set is used to estimate a normal distribution $\mathcal{N}(\mu, \Sigma)$ ($\mathcal{N}(\mu, \sigma)$ for univariate data) of the reconstruction error for multivariate input data. And the likelihood of a data point being abnormal is defined by the anomaly score

$$a_t = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} e^{-(e_t-\mu)^2/2\sigma^2} & d = 1 \\ (e_t - \mu)^T \Sigma^{-1} (e_t - \mu) & d > 1 \end{cases} \tag{7.4}$$

The largest anomaly score is picked up to represent the window anomaly score

$$a_{t+1}^{t+L} = \underset{i=1,\ldots,L}{max}(a_{t+i}) \tag{7.5}$$

In our work, we do not specify a threshold over the window anomaly scores to get a binary prediction. Instead, we directly evaluate the AUROC (Area Under the Receiver Operating Characteristic curve) score based on the real-valued anomaly scores. Different existing thresholding techniques can be applied to get a binary prediction in such a situation [MRA+16, SZN+19, HCL+18].

Based on the anomaly detection model above, we introduce a prototype layer between the encoder and decoder, which leverages interpretable prototypes of the normal data during the end-to-end training process. The prototype layer does not influence the information flow from the encoder to the decoder, i.e., the only information the decoder gets from the encoder is the last encoder's hidden state. The prototype layer contains k prototypes $p_i \in \mathbb{R}^m (i = 1...k)$ to be learned, where k is a user-defined parameter and k vectors are randomly initialized within the range $[-1, 1]$. As introduced in Section 7.3.3, several regularization terms are employed in the objective function to get the expected prototypes.

In most existing prototype-based models for classification task [MXQR19, KKK16, GGOGP19], the prototype layer is followed by some linear layers and a Softmax layer for the production of prediction, which increases the complexity of the network and requires additional regularization to enforce interpretability. As an anomaly detection model, our outputs are derived directly from the Autoencoder reconstruction errors. Therefore, we omit the additional layers after the prototype layer to simplify the network structure.

7.3.3 Objective function

The objective in the training phase is to 1) train the Autoencoder with normal windows such that the reconstruction error is minimized and 2) learn a batch of prototypes from the normal data. The reconstruction error loss of the Autoencoder is given by

$$\mathcal{L}_e = \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} e_{t+l} \tag{7.6}$$

where n is the number of sliding windows. To ensure that the learned prototypes are informative and diverse enough to each other, we use the diversity loss,

$$\mathcal{L}_d = \sum_{i=1}^{k} \sum_{j=i+1}^{k} max(0, d_{min} - ||p_i - p_j||_2^2)^2 \tag{7.7}$$

which defines the threshold d_{min} that only applies this penalize to nearby prototype pairs. Finally, to ensure the prototypes are representative of the local hidden representations, we define the following representation regularization term

$$\mathcal{L}_r = \frac{1}{k} \sum_{j=1}^{k} \underset{i \in [1,n]}{min} ||p_j - h_i||^2 + \frac{1}{n} \sum_{i=1}^{n} \underset{j \in [1,k]}{min} ||h_i - p_j||^2 \tag{7.8}$$

Table 7.1: Dataset description

	Length	Dimensionality	Anomaly rate (%)
Synthetic	20000	1	1.0
Taxi	17520	1	4.4
SMAP	562800	25	13.1
MSL	132046	55	10.7
SMD	56958	38	9.5

The first term encourages that each prototype is close to at least one hidden representation, while the second term encourages each hidden representation to be assigned to one prototype.

The overall objective function is

$$\mathcal{L} = \lambda_e \mathcal{L}_e + \lambda_d \mathcal{L}_d + \lambda_r \mathcal{L}_r \tag{7.9}$$

where λ_e, λ_d and λ_r are weighting hyperparameters.

7.4 Experiments

In this section, we introduce the experiments on ProtoAD under different settings. We experiment with different real-world datasets with a variety of anomaly types. In addition, to evaluate the model performance on specific data characteristics, we also introduce a synthetic dataset with artificial anomalies. Finally, we demonstrate the prototypes visually and analyze the prototype properties w.r.t. a variety of parameter settings. The source code of ProtoAD is available online [1].

7.4.1 Experiment setup

7.4.1.1 Dataset description

We experiment on one synthetic dataset and four common real-world benchmark datasets in the TSAD domain. The dataset properties are summarised in Table 7.1.

[1] https://github.com/KDD-OpenSource/ProtoAD

To understand the anomaly detection process and the learned prototypes, we introduce a one-dimensional synthetic dataset sampled from a sine wave with amplitude 1 and period 100 timestamps. A random noise $\epsilon \in [0, 0.1]$ is added to every timestamp. In addition, we multiply a random factor $\alpha \in [0, 1]$ to every $100th$ timestamp in the test set to simulate point anomalies. We define a half period of the sine wave as the window length (i.e., $L = 50$), such that the model is supposed to learn the crests and troughs as two types of prototypes.

The New York City Taxi (Taxi) dataset is a one-dimensional real-world dataset with a clear periodical feature. It records the passenger counts over days in 2014. Extreme passenger counts on public holidays are considered anomalies. Following [CSAH16], we aggregate the count numbers into 30-minute intervals. We take one day (i.e., $L = 48$) as the window length. SMAP and MSL are introduced in Section 2.4.1.1. They contain both point and contextual anomalies. However, anomaly data is also present in the train sets, which can impact the purity of prototypes. There is no common repeating pattern in these datasets. We set the window length L as 100. SMD is introduced in Section 4.4.1.1. We only use the data from one machine (machine-1-1) in our experiments. We set $L = 100$ for SMD.

7.4.1.2 Competitors

To the best of our knowledge, this is the first work that engages TSAD and prototype learning. The existing prototype learning networks [MXQR19, NCC+21, GGOGP19] commonly work in a supervised manner, which requires labeled data for the training phase. Therefore, they are not directly relevant to our setting. We mainly compare our method with the unsupervised anomaly detection approaches. Firstly, we compare with the LSTM-Autoencoder (LSTM-AE) [MRA+16], which has a similar setting as ours but without the prototype layer. Thereby, we can determine whether the prototype learning damages the original reconstruction-based anomaly detection. Furthermore, we compare with one of the state-of-the-art unsupervised TSAD approach OmniAnomaly [SZN+19]. We follow most of the default hyperparameter settings in [SZN+19] but use the window length same as in our work for the sliding window.

7.4.1.3 Evaluation metric

We adopt the AUROC score as the evaluation metric. Considering the essential requirement of detecting both point and contextual anomalies, we only evaluate on the window level. A data window is abnormal if it contains one or multiple abnormal instance(s).

Table 7.2: Anomaly detection performance (AUROC)

	LSTMAE	OmniAnomaly	ProtoAD (Ours)
Synthetic	0.50	0.95	0.54
Taxi	0.53	0.52	0.63
SMAP	0.41	0.49	0.40
MSL	0.73	0.50	0.73
SMD	0.95	0.51	0.95

7.4.1.4 Parameter configuration

In all experiments, we set $\lambda_e = 0.025$, $\lambda_d = 0.2$ and $\lambda_r = 0.5$. During training, 25% of the data is used for learning the parameters μ and Σ (σ for univariate data). All models are trained for 100 epochs with batch size 20, learning rate 0.0001, and dropout rate 0.2. We use the Adam optimizer [KB14]. All experiments are conducted on an NVIDIA Quadro RTX 6000 24GB GPU. The experimental results are averaged over three runs.

7.4.2 Performance

7.4.2.1 Overall performance

Firstly, we report the AUROC score over different models in Table 7.2. For ProtoAD, we take the number of prototypes $k = 10$. There is no significant difference between LSTMAE and ProtoAD, which indicates that the additional prototype layer and corresponding learning process do not directly impact the anomaly detection performance. ProtoAD even benefits from the prototype learning in the Synthetic and Taxi datasets. OmniAnomaly shows worse AUROC scores in comparison with the other two models. Different from [SZN+19], where all possible thresholds over the predicted anomaly scores are traversed, and only the threshold with the best $F1$ score is reported, the AUROC score reflects the more general quality of the anomaly scores over multiple thresholds.

7.4.2.2 Latent space visualization

In this section, we investigate a visualization of the Autoencoder hidden space to understand how time series data windows are embedded and how prototypes of normal data are learned. We use UMAP [MHM18] to reduce the high-dimensional latent representations into two dimensions. The result is visualized in Figure 7.2. Here, we set $k = 5$ for all datasets. The prototypes shown in the plots are learned during the training phase. The plotted normal and anomaly points are latent space embeddings of the test data.

In the synthetic data, the normal data lie in two regions. Four prototypes are learned from the trough half (lower left) and one from the crest half (middle right). In the real-world datasets, especially the SMAP and MSL with polluted training data, normal and abnormal data do not clearly show separated clusters. However, the learned prototypes represent the major blocks of dense regions showing normal patterns. Specifically, the prototypes gather at the bottom right corner for SMD, while no prototype is at the larger upper cluster. A possible reason is that the high-dimensional server data contains many zero values. The model can not summarize informative patterns in the train set, and slightly different normal patterns in the test data are mapped into a different region.

7.4.2.3 Prototype-based explanation

Finally, we map the prototypes learned in the latent space back to the human-interpretable input space. Similar to [CLT+18, MXQR19], we map the prototypes back to the input space using the nearest training data embedding in the latent space to prevent unrealistic produced by the decoder. Moreover, each neighbor can only be used once, making every prototype unique. We visualize the prototypes learned in the one-dimensional datasets Taxi and Synthetic in Figure 7.3 with five prototypes ($P1$ to $P5$) for each dataset.

In Figure 7.3a and Figure 7.3b, four similar prototypes ($P1$, $P2$, $P4$, $P5$) show an increasing taxi usage pattern in the morning and turning down at night. $P3$ can be seen as a delayed version of the other four, which is a weekend pattern. The light lines in the background are the normal (gray) and anomaly (red) sequences with the smallest distance to the corresponding prototypes in the latent space. Most of the normal patterns fit the assigned prototypes. A considerable number of both normal and anomaly sequences have the smallest latent space distance to $P3$. Some of them visually fit better with other prototypes, while the distance comparison and prototype assignment do not directly take place in the input space but in the latent

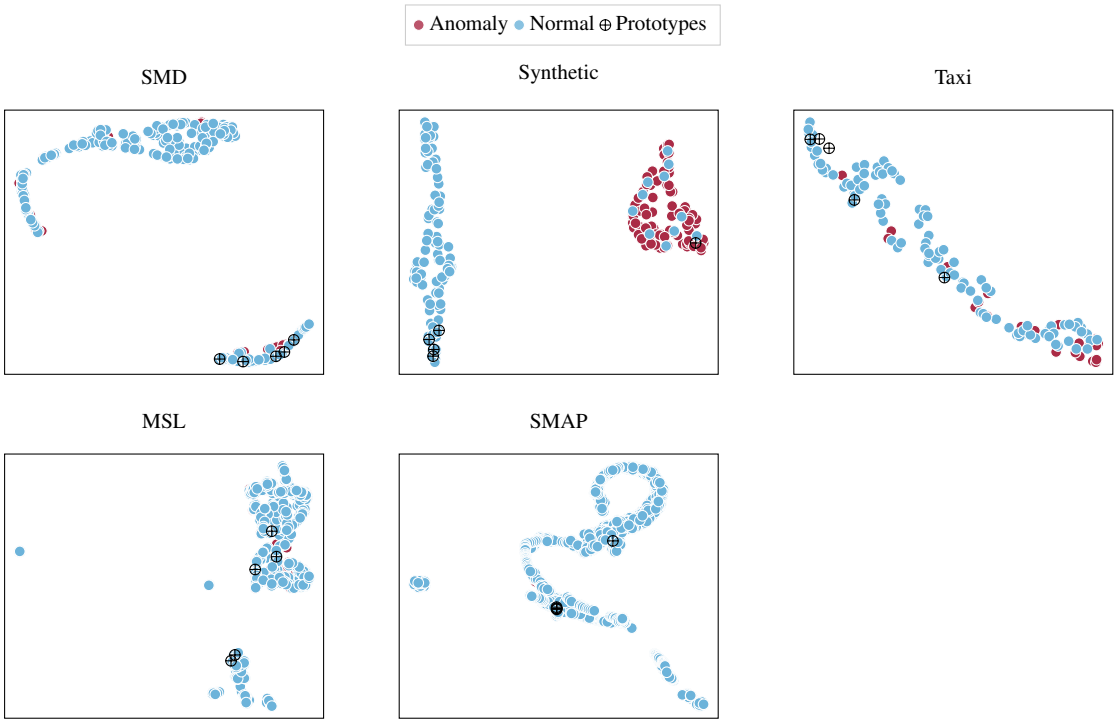


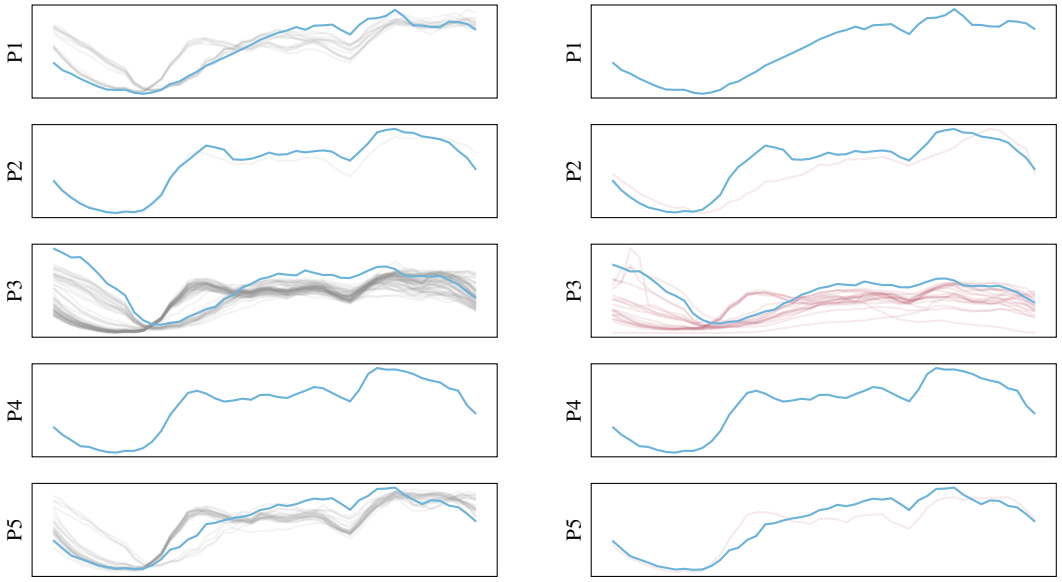
Figure 7.2: ProtoAD latent space visualization using UMAP

space. However, this is effective for long and high-dimensional sequences. Figure 7.3b depicts the explanation of anomaly patterns, namely how different are the anomaly sequences to their nearest prototypes. Figure 7.3c shows the three assigned normal and anomaly sequences (if available) to each prototype. Since the point anomalies are always generated at the beginning of the crest half, all anomalies are assigned to the crest prototype $P4$.

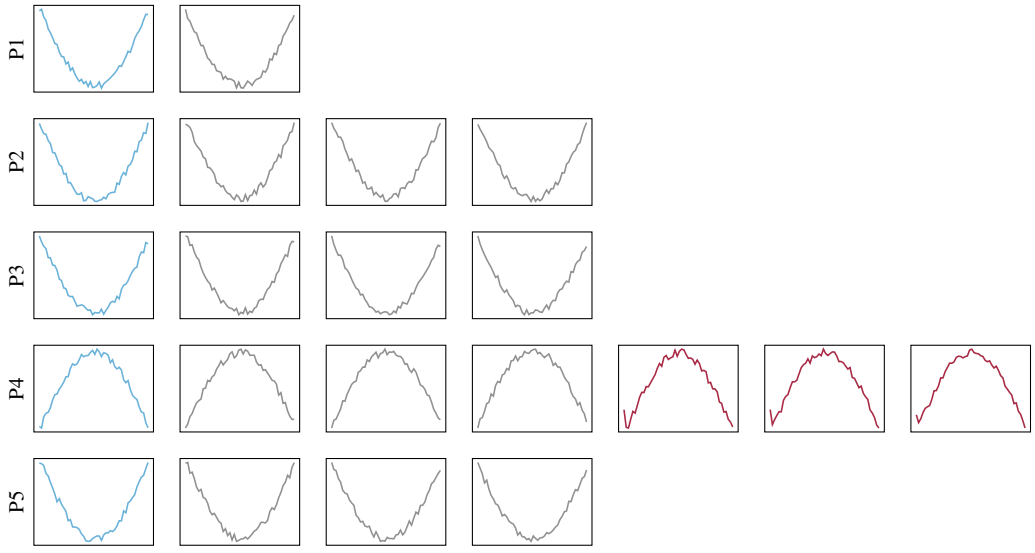
For the high-dimensional datasets, oftentimes we are only able to observe the prototypes in the latent space. In order to simplify the high-dimensional data in the input space, approaches in Chapter 5 can be used to extract the most important time period and features. Another extension possibility is to reduce the potential redundant prototypes (e.g., $P1$, $P2$, $P3$, $P5$ in Figure 7.3c).

7.4.2.4 Efficiency comparison

Training the Autoencoder with an extra prototype layer does not bring much training expense. We compare the epoch training time between LSTMAE ($k = 0$) and ProtoAD ($k \in$



(a) Taxi normal **(b)** Taxi anomaly



(c) Synthetic normal and anomaly

Figure 7.3: Prototype visualization (blue) with assigned normal (gray) and anomaly sequences (red).

$[5, 10, 20, 30, 50]$). As shown in Figure 7.4, there is no significant increase in training time for ProtoAD. On the contrary, due to the complex model structure, the epoch training time for

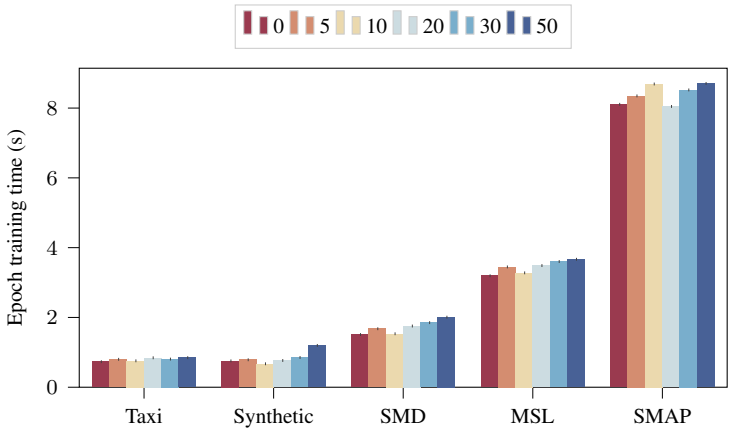


Figure 7.4: Efficiency analysis: epoch training time with different numbers of prototypes $k \in \{0, 5, 10, 20, 30, 50\}$.

OmniAnomaly is: Taxi $39s$, Synthetic $32s$, SMD $225s$, MSL $888s$ and SMAP $3627s$.

7.5 Conclusion

In this chapter, we explored using prototypes to explain reconstruction-based anomaly detection. Specifically, we integrate the recent end-to-end prototype learning into the LSTM-Autoencoder. We use the latent space representations in the Autoencoder, which are not directly used in the conventional reconstruction-based anomaly detection models. In our empirical evaluation, we figured out that adding a prototype learning step during the end-to-end training of the Autoencoders does not damage the performance of the Autoencoder. The prototypes contribute to an intuitive understanding of the normal patterns.

Although the prototypes learned in the two one-dimensional datasets are realistic and interpretable for humans, there are still two challenges. Firstly, the selection of parameter k is sometimes tricky. Pruning techniques can be applied to reduce the redundancy in the prototypes. Moreover, extracting useful information from high-dimensional or very long prototype sequences may be challenging. The global and local importance analysis introduced in Chapter 5 can be used as a prior step to let the visualization only focus on important features and time periods.

Part V

Conclusion

8

Conclusion

8.1 Discussion

Time series are ubiquitous, far beyond the common application areas. For example, treating the trajectory of an electron as a spatial-temporal sequence and viewing the evolution of bacterial community with environmental variables as a time series. Even the migration patterns in demographic changes constitute a time series. Among various time series analysis tasks, anomaly detection is one of the most essential and impactful. Identifying outliers within time series data contributes significantly to extracting informative knowledge within research contexts. However, a notable gap persists between cutting-edge research and the practical deployment and decision-making processes in real-world applications. This discrepancy highlights vital challenges from three distinct perspectives:

1. **Complex and noisy time series in real-world applications:** Time series collected in real-world scenarios are usually noisy and contain missing values. It is hard to identify task-relevant features and periods when collecting the data without deep domain knowledge and prior understanding of the machine learning models. This ambiguity leads to

significant redundancy. Moreover, due to the large volume of time series yet infrequent appearance of anomalies, labeling becomes extremely expensive.

2. **Real-time streaming processing:** The value of time series analysis usually finds a place in the real-time data generation and collection phase. Addressing anomalies at the earliest possible moment maximizes its impact. Consequently, models need to learn dynamic patterns with the continuous arrival of streaming data instances and adapt efficiently to the latest patterns.

3. **Temporal anomaly interpretation** A substantial gap exists between powerful deep anomaly detectors and the trustworthiness demand in safety-crucial application domains. While developing new anomaly detection models, the model interpretability and anomaly reasoning should be prioritized on par with the detection accuracy. Despite the recent advancements of interpretable machine learning techniques in the vision and text domain, efficiently and reliably interpreting complex time series data remains a persistent challenge.

In this dissertation, we first systematically introduced the background and existing solutions in the temporal anomaly detection and interpretation field in Part I. Addressing the three challenges highlighted above, we proposed our solution in the following three parts (Parts II to IV) accordingly. In Figure 8.1, we append a technique axis to the problem axis in Figure 1.2, which forms a sparse matrix for the studied problems and used solution techniques. In the remainder of this section, we summarize our solution techniques and sketch the potential extension possibilities.

8.1.1 Time series anomaly detection

In Chapter 2, we proposed ContrastAD, a contrastive learning-based anomaly detection approach for static time series. The representation learning of complex time series is an important bottleneck in improving anomaly detection performance. Inspired by recent advances in contrastive learning in the image representation learning field, we extend it to the time series domain. A major challenge is to define contrastive pairs for the learning procedure. Common image data transformations (e.g., cropping, rotation) for positive sample generation are not ideal for time series data due to their temporal nature. To this end, we proposed anomaly-guided artificial transformation for time series data. Specifically, we add random noise to the

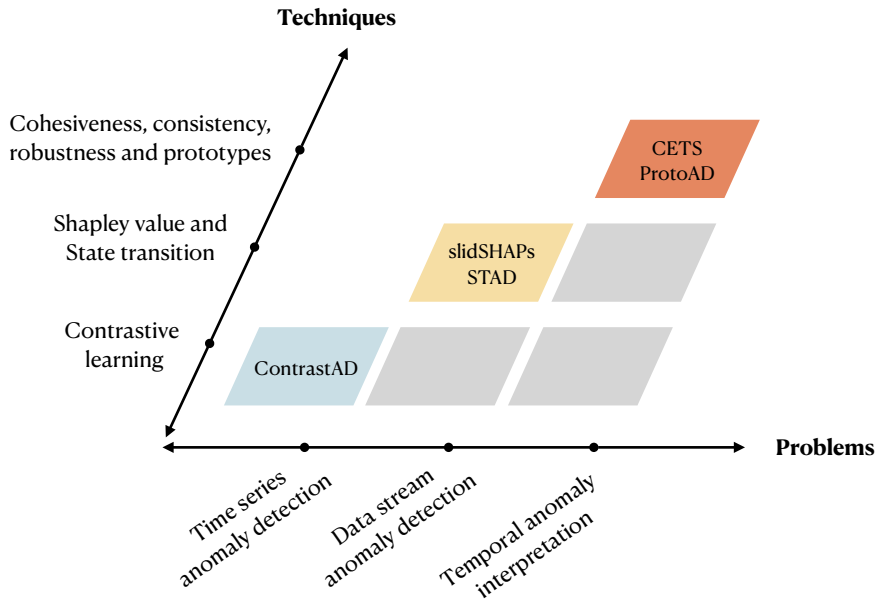


Figure 8.1: Problem-technique matrix: a sparse matrix demonstrating the studied problems as well as used solution techniques (colored cells). The gray-shaded cells denote potential future work directions.

target time window to generate positive transformation while applying multiple artificial point and contextual anomalies as negative transformations. With extensive experiments on standard benchmark datasets, we showed supreme performance of the proposed method and efficacy of the artificial temporal transformations.

Beyond our work, we still see promising potential for contrastive learning on data stream analysis and time series interpretation. Thanks to the unsupervised representation learning capability, contrastive learning may contribute to drift detection or be applied for adaptive online anomaly detection. Furthermore, from the interpretability perspective, the positive and negative pairs in contrastive learning are analogous to example-based interpretations, e.g., prototypes and counterfactuals. Therefore, the contrastive learning outcomes could be explicitly or implicitly used as interpretations.

8.1.2 Data stream anomaly detection

When applying an anomaly detector in the online setting, the primary challenge is the ever-changing pattern in the data stream. Hence, in Chapter 3, we investigated slidSHPAs, a novel unsupervised drift detection approach. Unlike a major category of drift detection approaches

using labels and online prediction errors, we seriously consider the lack of labels in real-world applications and focus on the prior distribution drifts in the data stream. Specifically, we proposed to detect the correlation drifts of categorical features indicated by the total correlation. We extend the common usage of Shapley values in the feature attribution analysis to indicate correlation drifts, where total correlation is the value function of Shapley values. This enables us to measure drifts in any possible subspaces systematically. In our empirical evaluation, we first qualitatively visualized the effect of visualizing correlation drifts in the slidSHAPs space and then quantitatively validated the performance advance to classical drift detectors using other statistical descriptions of the data stream.

In Chapter 4, we proposed STAD, an online state-transition-aware anomaly detection framework for effectively maintaining and adapting the models. Inspired by the automata theory in theoretical computer science, we modeled each data distribution (i.e., each concept in the concept drift detection context) as well as the model learned from that data distribution as a state. We modeled the concept drift in data streams as state transitions, which enabled us to efficiently compare similarities between states and potentially reuse existing models from similar states to reduce the online model retraining effort. With empirical evaluation, we figured out that the proposed approach outperforms standard time series and streaming anomaly detectors. The explicit modeling of state transition also increases human understanding of the data stream evolution as well as model scheduling.

We use Shapley values as a feature descriptor of data streams; however, they are more commonly used for feature attribution analysis. It is interesting to incorporate the interpretation aspect of Shapley values into the current framework. For example, feature attribution can be added as additional information to describe a state (in addition to the data distribution and Autoencoder model).

8.1.3 Temporal anomaly interpretation

In Chapters 5 and 6, we addressed the challenges of directly applying existing saliency methods for image interpretation to time series data. Firstly, in Chapter 5, we investigated the cohesiveness of time series predictions. Targeting simplified and concentrated saliency maps for better human understanding, we proposed a two-step global-local perturbation-based attribution analysis using time series prototypes. Using pre-learned prototypes, we ensure the realisticness of time series perturbations. By sequentially applying the global and local attribution analysis, we reduced and concentrated the salient regions in the saliency maps. We empirically showed

the outstanding attribution quality of the proposed approach to non-time series saliency approaches. Furthermore, we also qualitatively visualized the cohesive explanations generated by our approach.

In Chapter 6, we further investigated the necessary properties of time series attributions between adjacent sliding windows. Due to the contextual dependency of time series, a necessary property of their feature attribution visualization is temporal consistency; namely, the same timestamp should show consistent relative attribution in adjacent overlapping sliding windows. In contrast, the features of time series do not implicate particular ordering; therefore, robust attribution should persist in relative feature attribution even by feature swapping. We evaluated existing saliency methods for general purposes and figured out their weakness in time series predictions with regard to consistency and robustness.

Considering the rare abnormal event in massive time series, we claim an example-based explanation is one of the most intuitive ways to provide reasoning for an anomaly. As a showcase, in Chapter 7, we proposed a prototype-based approach to explain time series anomalies in the reconstruction-based models. Specifically, we effectively learned representative and distinct samples in the latent space of Autoencoders as prototypes of the normal data. Predicted anomalies are further visually compared with the prototypes for interpretation.

8.2 Future works

Despite our efforts in this research area, several aspects can be further improved. We emphasize them with four future research directions (RD).

RD1: State modeling using Markov Model

In Chapter 4, we proposed to use a state-transition model to capture drifts in the Autoencoder latent space for efficient model maintenance and adaptation. Similar research on concept modeling in sequential data also uses the Hidden Markov Model (HMM). HMM has a similar target to our approach STAD (Chapter 4). STAD is inspired by the automata theory and focuses on the identification of reoccurring concepts in the Autoencoder latent space by measuring the Kullback-Leibler divergence. HMM is a stochastic model that focuses on the modeling of state and transition probabilities between states. Common Markov models for data stream modeling works include Yang et al. [YWZ06] that use a Markov chain to model the state transition by

observing classifier performance change. Angel et al. [ABE16] use an HMM to predict concept reoccurrence with a state pool. Ahmadi et al. [AK18] use a first-order Markov chain to not only maintain the state transitions but also actively merge states to keep online efficiency. However, existing Markov models work majorly in the input space. Therefore, incorporating latent representation-based drift detection and state-matching STAD with the probabilistic state transition modeling may boost the efficiency of online streaming processing. Specifically, when state transition or model adaption of STAD is affected by a lack of data or a suspicion of polluted normal data, a pre-modeled Markov model may serve as an alternative solution.

RD2: Potentials in time series contrastive learning

In Chapter 2, we employed a contrastive learning-based approach for time series representation learning and anomaly detection. In the contrastive loss design, we target constructing contrastive pairs within mini-batches, i.e., minimizing distances between positive pairs and maximizing distances between negative pairs within each mini-batch. The proposed method shows validity on the tested time series datasets, while in other application domains [JBZ+20], more efficient designs of the contrastive loss are applied to large-scale datasets. In computer vision, image data essentially allows various valid representations of the target objects, e.g., photos from different perspectives with different poses. Hence, it is common to conduct contrastive learning on large benchmark datasets (e.g., CIFAR-10 [KH+09], ImageNet ILSVRC-2012 [RDS+15]). When conducting contrastive learning within mini-batches, in order to cover sufficient global information (i.e., comparing the target sample with as many other negative samples in the rest of the dataset as possible), the mini-batches are supposed to be defined as very large, e.g., default batch size 4096 in SimCLR [CKNH20]. Another popular alternative design is maintaining a memory bank [WXYL18], which contains latent embedding of all instances in the dataset. During the contrastive learning procedure, every sample can be operated with any other sample in the dataset, and the memory bank representation is updated continuously along with the encoder. In order to keep the representation consistency, He et al. [HFW+20] use a momentum approach to balance the new and old representations.

In time series, though the raw representations are also diverse, the difference between samples is more caused by random noise or stochastic permutations, while the essential statistical features (e.g., frequency, periodic, trend) underlying in the same class stay stable. Therefore, the mini-batch approach without a memory bank in Chapter 2 worked for our anomaly detection task. It is worth figuring out whether and how much the anomaly detection model can

benefit from the memory bank and how well a memory bank can efficiently represent the time series data and be updated during the representation learning.

RD3: Fair evaluation of anomaly detector

Both time series and data stream anomaly detection research often use the F1-score and AUROC (Area Under the Receiver Operating Characteristic curve) as evaluation metrics. However, these two metrics do not explicitly consider the temporal anomalies in sequential data and sometimes lead to biased performance results after aligning point-wise predictions to data windows. Specifically for contextual anomalies taking place over several timestamps, in recent adjusted evaluation approaches, the whole abnormal event subsequence is marked as true positive, as far as the anomaly detector alarms at any timestamp in between. Doshi et al. [DAY22] have shown that even random guesses sometimes outperform state-of-the-art models using the adjusted evaluation. Therefore, the research domain urgently needs a fair evaluation metric for time series and stream anomaly detection. Several aspects should be considered, including distinguishing between point and contextual anomalies, precise identification of abnormal event periods (potentially also sub-feature space), average alarm delay, and balance between false positives and false negatives.

RD4: Quantitative evaluation of time series interpretations

In Chapters 5 and 6, we elaborated the demand of making time series interpretations meaningful and human-understandable, i.e., cohesiveness, consistency, and robustness, however, not only in time series interpretation but also generally in the interpretable machine learning field, the quantification of interpretability is very challenging. This makes the practical design of interpretability approaches hard to evaluate and justify. Therefore, quantifiable interpretation for time series is a desired research direction in the upcoming decade.

Existing interpretability approaches are often implicitly evaluated. For example, in feature attribution analysis, an interpreter is supposed to be highly qualified if the top-ranked features lead to a significant performance drop when they get masked. These approaches, however, neglect the human-understandability aspect, e.g., cohesiveness and simplicity. Therefore, further explicit quantitative measurements are desired, especially for complex data like time series. One possible extension to CETS in Chapter 5 is to design a metric to quantify cohesiveness,

which should consider the feature attribution ranking, sparsity of saliency map, and contrastiveness of saliency cells as a whole. In order to incorporate subjective justification, human feedback is an inevitable step in interpretability evaluation. Case studies by crowdsourcing is an inexpensive yet efficient option [HLM12, LTY+21].

8.3 Practical impact

During our research, we also investigated real-world problems in the relevant domain. As a student project [MES+22], we have developed a web application for energy consumption data anomaly detection at the university campus. By monitoring data streams from multiple environmental sensors, the application can detect and interpret anomaly events. For example, broken windows may lead to continuous unnecessary heating in winter.

Targeting helping researchers in the anomaly detection field, we developed a visualization tool for reconstruction-based anomaly detectors [BLJM23]. Specifically, the latent space visualization and details of the training procedure contribute to efficient model development and understanding.

References

[AAAB18] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer, 2018. 22

[AALC21] Emre Ates, Burak Aksar, Vitus J Leung, and Ayse K Coskun. Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8. IEEE, 2021. 114

[ABE16] Abad Miguel Angel, Gomes Joao Bartolo, and Menasalvas Ernestina. Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function. *Expert Systems with Applications*, 46:87–105, 2016. 132

[AG12] Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*, pages 1–12, 2012. 11

[AGAA18] Hamoud Alshammari, Oussama Ghorbel, Mohammed Aseeri, and Mohamed Abid. Non-negative matrix factorization (nmf) for outlier detection in wireless sensor networks. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 506–511. IEEE, 2018. 7

[Agg16] Charu C Aggarwal. Outlier analysis second edition, 2016. 10

[AK18] Zahra Ahmadi and Stefan Kramer. Modeling recurring concepts in data streams: a graph-based framework. *Knowledge and Information Systems*, 55:15–44, 2018. 38, 39, 40, 65, 132

[ALPA17] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017. 9, 63, 72, 113

[AMSR21] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using Shapley Additive Explanations. *Expert Systems with Applications*, 2021. 41

[AN07] Arthur Asuncion and David Newman. Uci machine learning repository, 2007. 11

[APHW03] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003. 8

[BA95] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 1995. 46

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. 101

[BC21] Mark A. Burgess and Archie C. Chapman. Approximating the Shapley Value Using Stratified Empirical Bernstein Sampling. In *IJCAI*, 2021. 47

[BD99] Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999. 72

[BG07] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SDM*. SIAM, 2007. 15, 37, 40, 43, 51, 62, 64

[BGBMY01] Ana Maria Bianco, M Garcia Ben, EJ Martinez, and Vıctor J Yohai. Outlier detection in regression models with arima errors using robust estimates. *Journal of Forecasting*, 20(8):565–579, 2001. 63

[BGdCÁF+06] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006. 37, 39, 43

[BH95] 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)*, 1995. 46

[BH20] Liron Bergman and Yedid Hoshen. Classification-Based Anomaly Detection for General Data, May 2020. arXiv:2005.02359 [cs, stat]. 23

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010. 11

[BHMM22] Chiara Balestra, Florian Huber, Andreas Mayr, and Emmanuel Müller. Unsupervised Features Ranking via Coalitional Game Theory for Categorical Data. In *DaWaK*, 2022. 38, 39, 41, 43, 44, 47, 52, 59

[BKK18] 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. 20, 22, 30

[BKNS00] 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. 6, 20, 22, 30

[BLJM23] Arn Baudzus, Bin Li, Adnane Jadid, and Emmanuel Müller. The good, the bad, and the average: Benchmarking of reconstruction based multivariate time series anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 356–360. Springer, 2023. 134

[BLM23] Chiara Balestra, Bin Li, and Emmanuel Müller. On the consistency and robustness of saliency explanations for time series classification. *arXiv preprint arXiv:2309.01457*, 2023. 84

[BNZ22] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4488–4499, 2022. 5

[BPRH13] Albert Bifet, Bernhard Pfahringer, Jesse Read, and Geoff Holmes. Efficient data stream classification via probabilistic adaptive windows. In *Proceedings of the 28th annual ACM symposium on applied computing*, 2013. 50

[BSC+21] João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2021. 39, 41, 80, 83, 98, 100

[BSH+10] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010. 100, 101

[CBS+16] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems*, 29, 2016. 80, 83, 97, 101

[CCJ+20] Michelangelo Ceci, Roberto Corizzo, Nathalie Japkowicz, Paolo Mignone, and Gianvito Pio. Echad: embedding-based change detection from multivariate time series in smart grids. *IEEE Access*, 8:156053–156066, 2020. 63

[CDR07] Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature Selection via Coalitional Game Theory. *Neural Computing*, 2007. 41

[CEQZ06] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006. 8

[CF07] Mooi Choo Chuah and Fen Fu. Ecg anomaly detection via time series analysis. In *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops: ISPA 2007 International Workshops SSDSN, UPWN, WISH, SGC, ParDMCom, HiPCoMB, and IST-AWSN Niagara Falls, Canada, August 28-September 1, 2007 Proceedings 5*, pages 123–135. Springer, 2007. 5, 8

[CGT09] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 2009. 47

[CHR+15] Yanping Chen, Yuan Hao, Thanawin Rakthanmanon, Jesin Zakaria, Bing Hu, and Eamonn Keogh. A general framework for never-ending learning from time series streams. *Data mining and knowledge discovery*, 29(6):1622–1664, 2015. 26, 29

[CK22] Lucas Cazzonelli and Cedric Kulbach. Detecting anomalies with autoencoders on data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 258–274. Springer, 2022. 7

[CKH+15] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/. 11, 103

[CKL22] Ian Covert, Chanwoo Kim, and Su-In Lee. Learning to Estimate Shapley Values with Vision Transformers, 2022. 47

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 23, 132

[CLAM+21] Jacinto Carrasco, David López, Ignacio Aguilera-Martos, Diego García-Gil, Irina Markova, Marta Garcia-Barzana, Manuel Arias-Rodil, Julián Luengo, and Francisco Herrera. Anomaly detection in predictive maintenance: A new evaluation framework for temporal unsupervised anomaly detection algorithms. *Neurocomputing*, 462:440–452, 2021. 5, 8

[CLT+18] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018. 112, 114, 121

[CMO16] Rodolfo C Cavalcante, Leandro L Minku, and Adriano LI Oliveira. Fedd: Feature extraction for explicit concept drift detection in time series. In *IJCNN*, 2016. 39, 40

[Con99] William Jay Conover. *Practical nonparametric statistics*, volume 350. john wiley & sons, 1999. 46

[CS21] Jonathan Crabbé and Mihaela Van Der Schaar. Explaining Time Series Predictions with Dynamic Masks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2166–2177. PMLR, July 2021. ISSN: 2640-3498. 81, 83, 84, 90

[CSAH16] Yuwei Cui, Chetan Surpur, Subutai Ahmad, and Jeff Hawkins. A comparative study of htm and other neural network models for online sequence learning with streaming data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1530–1538. IEEE, 2016. 119

[CSL21] Hyunsoo Cho, Jinseok Seol, and Sang-goo Lee. Masked Contrastive Learning for Anomaly Detection. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1434–1441, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. 21, 23

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995. 7

[CV15] Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ecg time signals via deep long short-term memory networks. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–7. IEEE, 2015. 5, 8

[CWZ+16] Chao Chen, Yu Wang, Jun Zhang, Yang Xiang, Wanlei Zhou, and Geyong Min. Statistical features-based real-time detection of drifted twitter spam.

IEEE Transactions on Information Forensics and Security, 12(4):914–925, 2016. 67, 69

[CYPY21] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 2021. 10

[CZS+16] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data mining and knowledge discovery*, 30(4):891–927, 2016. 72

[DAA+19] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, 32, 2019. 100

[DAY22] Keval Doshi, Shatha Abudalou, and Yasin Yilmaz. Reward once, penalize once: Rectifying time series anomaly detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022. 133

[DBK+19] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. 89

[dCDVdM17] Fausto G da Costa, Felipe SLG Duarte, Rosane MM Vallim, and Rodrigo F de Mello. Multidimensional surrogate stability to detect data stream concept drift. *Expert Systems with Applications*, 2017. 40

[DF13] Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013. 8

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. 50

[DH21] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021. 11

[dHL21] Puck de Haan and Sindy Löwe. Contrastive predictive coding for anomaly detection. *arXiv preprint arXiv:2107.07820*, 2021. 23, 29

[DJ18] Yue Dong and Nathalie Japkowicz. Threaded ensembles of autoencoders for stream learning. *Computational Intelligence*, 34(1):261–281, 2018. 72

[DKVY06] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006. 40, 43, 46, 51, 67, 69

[DLZS17] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017. 5

[DP11] Gregory Ditzler and Robi Polikar. Hellinger distance based drift detection for nonstationary environments. In *CIDUE*, 2011. 40, 51

[dRFMB16] Denis Moreira dos Reis, Peter Flach, Stan Matwin, and Gustavo Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1545–1554, 2016. 46, 64

[dSAdSAF+21] Jesimar da Silva Arantes, Márcio da Silva Arantes, Herberth Birck Fröhlich, Laure Siret, and Renan Bonnard. A novel unsupervised method for anomaly detection in time series based on statistical features for industrial predictive maintenance. *International journal of data science and analytics*, 12(4):383–404, 2021. 5, 8

[DWP+22] Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu C Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *arXiv preprint arXiv:2211.05244*, 2022. 10

[EJS+19] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott M Lundberg, and Su-In Lee. Learning explainable models using attribution priors. 2019. 89

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 20

[Eng23] Joseph Enguehard. Learning perturbations to explain time series predictions. *arXiv preprint arXiv:2305.18840*, 2023. 83

[ERC+21] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*, 2021. 30

[FBdCÁRJ+14] Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yaile Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014. 39, 49

[FPV19] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2950–2958, 2019. 83

[FRC+19] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489, 2019. 8

[FRD19] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *J. Mach. Learn. Res.*, 20(177):1–81, 2019. 82

[FV17] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017. 82, 83

REFERENCES

[Gam10] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010. 10

[GCGDS20] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda Dos Santos. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2020. 40

[GEY18] Izhak Golan and Ran El-Yaniv. Deep Anomaly Detection Using Geometric Transformations, November 2018. arXiv:1805.10917 [cs, stat]. 21, 23, 30

[GGOGP19] Alan H Gee, Diego Garcia-Olano, Joydeep Ghosh, and David Paydarfar. Explaining deep classification of time-series data with learned prototypes. In *CEUR workshop proceedings*, volume 2429, page 15. NIH Public Access, 2019. 84, 86, 112, 114, 117, 119

[GMCR04] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004. 15, 37, 39, 40, 62

[GMS+20] Riccardo Guidotti, Anna Monreale, Francesco Spinnato, Dino Pedreschi, and Fosca Giannotti. Explaining any time series classifier. In *CogMI*. IEEE, 2020. 41

[GRN18] Ralf Greis, T Reis, and C Nguyen. Comparing prediction methods in anomaly detection: an industrial evaluation, 2018. 22

[GŽB+14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014. 115

[GZZ+23] Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. Anomalygpt: Detecting industrial anomalies using large vision-language models. *arXiv preprint arXiv:2308.15366*, 2023. 9

[Haw80] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980. 3

[HCL+18] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018. 5, 11, 22, 29, 50, 64, 67, 116

[HCL21] Xu Han, Xiaohui Chen, and Li-Ping Liu. Gan ensemble for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4090–4097, 2021. 9

[HCLR19] Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. Interpretable image recognition with hierarchical prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 32–40, 2019. 112, 114

[Hen20] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020. 29

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009. 11

[HFW+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 132

[HHH+22] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35:32142–32159, 2022. 6, 8, 11

[HKR+21] Ben Halstead, Yun Sing Koh, Patricia Riddle, Mykola Pechenizkiy, Albert Bifet, and Russel Pears. Fingerprinting concepts in data streams with supervised and unsupervised meta-information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1056–1067. IEEE, 2021. 39, 40, 48

[HLM12] Amanda Hutton, Alexander Liu, and Cheryl Martin. Crowdsourcing evaluations of classifier interpretability. In *2012 AAAI Spring Symposium Series*, 2012. 134

[Ho95] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995. 7

[Ho05] Shen-Shyang Ho. A martingale framework for concept change detection in time-varying data streams. In *ICML*, 2005. 50

[Hof07] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007. 20

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 100, 103

[HW22] Siho Han and Simon S Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2977–2986, 2022. 11

[HWT+15] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015. 80

[HYT23] Xiao Han, Shuhan Yuan, and Mohamed Trabelsi. Loggpt: Log anomaly detection via gpt. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1117–1122. IEEE, 2023. 9

[HZ19] Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series*, volume 1213, page 042050. IOP Publishing, 2019. 113

[IGCBF20] Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. *Advances in neural information processing systems*, 33:6441–6452, 2020. 10, 11, 80, 81, 82, 98, 101, 103, 104

[IVML18] Philokypros Ioulianou, Vasileios Vasilakis, Ioannis Moscholios, and Michael Logothetis. A signature-based intrusion detection system for the internet of things. *Information and Communication Technology Form*, 2018. 5

[JBZ+20] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020. 20, 132

[JW19] Sarthak Jain and Byron C. Wallace. Attention is not explanation. *CoRR*, abs/1902.10186, 2019. 97

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 30, 90, 120

[Ken48] Maurice George Kendall. Rank correlation methods. 1948. 104

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 132

[KKK16] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016. 117

[KKSZ09] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pages 831–838. Springer, 2009. 9

[KLF05] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *ICDM*, 2005. 59, 103

[KMB12] Fabian Keller, Emmanuel Muller, and Klemens Bohm. Hics: High contrast subspaces for density-based outlier ranking. In *2012 IEEE 28th international conference on data engineering*, pages 1037–1048. IEEE, 2012. 9, 38, 39

[KMDC17] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017. 83

[KML20] Patrick Kidger, James Morrill, and Terry Lyons. Generalised interpretable shapelets for irregular time series. *arXiv preprint arXiv:2005.13948*, 2020. 114

[KMM+20] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020. 11

[KR09] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009. 86

[KS20] Pooja Kamat and Rekha Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S web of conferences*, volume 170, page 02007. EDP Sciences, 2020. 5, 8

[KTNA21] Eamonn Keogh, Dutta Roy Taposh, U Naik, and A Agrawal. Multi-dataset time-series anomaly detection competition. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://compete. hexagonml. com/practice/competition/39*, 2021. 29

[KZK+19] Deepak A Kaji, John R Zech, Jun S Kim, Samuel K Cho, Neha S Dangayach, Anthony B Costa, and Eric K Oermann. An attention based deep learning model of clinical events in the intensive care unit. *PloS one*, 14(2):e0211057, 2019. 101

[LBM22] Bin Li, Chiara Balestra, and Emmanuel Müller. Enabling the visualization of distributional shift using shapley values. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022. 39, 41

[LCX+20] Guozhong Li, Byron Koon Kau Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace LH Wong. Efficient shapelet discovery for time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 2020. 114

[LEC+19] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable ai for trees: From local explanations to global understanding. *arXiv*, 2019. 54

[LFV+17] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017. 100, 103

[LJHR23] Victor Livernoche, Vineet Jain, Yashar Hezaveh, and Siamak Ravanbakhsh. On diffusion modeling for anomaly detection. *arXiv preprint arXiv:2305.18593*, 2023. 9

[LJM23] Bin Li, Carsten Jentsch, and Emmanuel Müller. Prototypes as explanation for time series anomaly detection. *arXiv preprint arXiv:2307.01601*, 2023. 81, 84, 86

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 41, 83, 98, 100

[LLCR18] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 84, 112, 113, 114

[LLD+18] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018. 10, 15, 38, 62, 63, 64, 65, 66

[LLP07] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007. 6

[LMA23] Daesoo Lee, Sara Malacarne, and Erlend Aune. Explainable anomaly detection using masked latent generative modeling. *arXiv preprint arXiv:2311.12550*, 2023. 10

[LRS+23] Kin Kwan Leung, Clayton Rooke, Jonathan Smith, Saba Zuberi, and Maksims Volkovs. TEMPORAL DEPENDENCIES IN FEATURE IMPORTANCE FOR TIME SERIES PREDICTION. 2023. 80, 83, 90, 91

[LTY+21] Xiaotian Lu, Arseny Tolmachev, Tatsuya Yamamoto, Koh Takeuchi, Seiji Okajima, Tomoyoshi Takebayashi, Koji Maruhashi, and Hisashi Kashima. Crowdsourcing evaluation of saliency-based xai methods. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part V 21*, pages 431–446. Springer, 2021. 134

[LTZ08] 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. 7, 20, 30

[LZVL23] Zhong Li, Yuxuan Zhu, and Matthijs Van Leeuwen. A survey on explainable anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 18(1):1–54, 2023. 11

[LZW+21] Kwei-Herng Lai, Daochen Zha, Guanchu Wang, Junjie Xu, Yue Zhao, Devesh Kumar, Yile Chen, Purav Zumkhawaka, Minyang Wan, Diego Martinez, and Xia Hu. Tods: An automated time series outlier detection system. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(18):16060–16062, May 2021. 11

[LZWW23] Zhenzhen Liu, Jin Peng Zhou, Yufan Wang, and Kilian Q Weinberger. Unsupervised out-of-distribution detection with diffusion inpainting. *arXiv preprint arXiv:2302.10326*, 2023. 9

[LZX+21] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*, 2021. 6, 11

[MCFH22] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. *The Journal of Machine Learning Research*, 23(1):2082–2127, 2022. 47, 83

[MES+22] Benedikt Tobias Müller, Marvin Ender, Jan Erik Swiadek, Mengcheng Jin, Simon Winkel, Dominik Niedziela, Bin Li, Jelle Hüntelmann, and Emmanuel Müller. Adept: Anomaly detection, explanation and processing for time series with a focus on energy consumption data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 622–626. Springer, 2022. 134

[MFPB10] Stefano Moretti, Vito Fragnelli, Fioravante Patrone, and Stefano Bonassi. Using coalitional games on biological networks to measure centrality and power of genes. *Bioinformatics*, 2010. 41

[MHM18] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 121

[MHM+21] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. River: machine learning for streaming data in python. 2021. 11

[MLA18] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1963–1972, 2018. 9

[Mol22] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. 10, 83

[MRA+16] 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. 20, 22, 30, 62, 64, 67, 113, 115, 116, 119

[MRBA18] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018. 11

[MT16] Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016. 5, 29

[MVS+15] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pages 89–94, 2015. 20, 22, 64, 72

[MVW+15] Erik Marchi, Fabio Vesperini, Felix Weninger, Florian Eyben, Stefano Squartini, and Björn Schuller. Non-linear prediction with lstm recurrent neural networks for acoustic novelty detection. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015. 62, 67

[MXQR19] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 903–913, 2019. 112, 114, 117, 119, 121

[MZH16] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European conference on computer vision*, pages 527–544. Springer, 2016. 23

[MZLZ19] Hengyu Meng, Yuxuan Zhang, Yuanxiang Li, and Honghua Zhao. Spacecraft anomaly detection via transformer reconstruction error. In *International Conference on Aerospace System Science and Engineering*, pages 351–362. Springer, 2019. 64

[NCC+21] Jingchao Ni, Zhengzhang Chen, Wei Cheng, Bo Zong, Dongjin Song, Yanchi Liu, Xuchao Zhang, and Haifeng Chen. Interpreting convolutional sequence model by learning local prototypes with adaptation regularization. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1366–1375, 2021. 112, 114, 119

[NLD+19] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In *CNS*, 2019. 41

[NTP+23] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Computing Surveys*, 55(13s):1–42, 2023. 84

[OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 21, 23, 29, 30

[PDS18] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018. 82

[Pev16] T. Loda Pevný. Loda: Lightweight on-line detector of anomalies, 2016. 9

[PHC21a] Qingyi Pan, Wenbo Hu, and Ning Chen. Two Birds with One Stone: Series Saliency for Accurate and Interpretable Multivariate Time Series Forecasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 2884–2891, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. 83

[PHC21b] Qingyi Pan, Wenbo Hu, and Ning Chen. Two birds with one stone: Series saliency for accurate and interpretable multivariate time series forecasting. In *IJCAI*, pages 2884–2891, 2021. 98

[PHHN16] Karlson Pfannschmidt, Eyke Hüllermeier, Susanne Held, and Reto Neiger. Evaluating Tests in Medical Diagnosis: Combining Machine Learning with Game-Theoretical Concepts. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. 2016. 41

[PKO+22] Vipin Pillai, Soroush Abbasi Koohpayegani, Ashley Ouligian, Dennis Fong, and Hamed Pirsiavash. Consistent explanations by contrastive learning. In *CVPR*, 2022. 101

[PKS18] Randy Paffenroth, Kathleen Kay, and Les Servi. Robust pca for anomaly detection in cyber networks. *arXiv preprint arXiv:1801.01571*, 2018. 22

[PLL07] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. Incremental local outlier detection for data streams. In *2007 IEEE symposium on computational intelligence and data mining*, pages 504–515. IEEE, 2007. 8

[PS19] João Pereira and Margarida Silveira. Unsupervised representation learning and anomaly detection in ecg sequences. *International Journal of Data Mining and Bioinformatics*, 22(4):389–407, 2019. 5, 8

[PSvdH19] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 353–362, 2019. 7, 8

[PV16] Ali Pesaranghader and Herna L Viktor. Fast hoeffding drift detection method for evolving data streams. In *ECML PKDD*, 2016. 51

[PVP18] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. Mcdiarmid drift detection methods for evolving data streams. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2018. 49, 51, 64, 71

[PZX+18] Kai Peng, Lixin Zheng, Xiaolong Xu, Tao Lin, and Victor CM Leung. Balanced iterative reducing and clustering using hierarchies with principal component analysis (pbirch) for intrusion detection over big data in mobile cloud

environment. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage: 11th International Conference and Satellite Workshops, SpaCCS 2018, Melbourne, NSW, Australia, December 11-13, 2018, Proceedings 11*, pages 166–177. Springer, 2018. 8

[QAWZ15] Abdulhakim A Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *KDD*, 2015. 38, 39, 40, 48, 50

[QPK+22] Chen Qiu, Timo Pfrommer, Marius Kloft, Stephan Mandt, and Maja Rudolph. Neural Transformation Learning for Deep Anomaly Detection Beyond Images, February 2022. arXiv:2103.16440 [cs]. 21, 23, 28, 30

[Ray16] Shebuti Rayana. ODDS library, 2016. 11

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 132

[RGL19] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019. 46, 64

[RK05] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 506–510. SIAM, 2005. 86

[RKV+21] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. 11

[RL05] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005. 20

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *KDD*, 2016. 41, 80, 82, 98

[RVG+18] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018. 7, 8

[RVG+19] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019. 8, 22

[RWB+22] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The Shapley Value in Machine Learning. *IJCAI*, 2022. 41

[SAEA+19] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A Keim. Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE, 2019. 82, 100

[SBS+17] Arvind Kumar Shekar, Tom Bocklisch, Patricia Iglesias Sánchez, Christoph Nikolas Straehle, and Emmanuel Müller. Including multi-feature interactions and redundancy for feature ranking in mixed datasets. In *ECML PKDD*, 2017. 38, 39

[SCM21] Vikash Sehwag, Mung Chiang, and Prateek Mittal. SSD: A Unified Framework for Self-Supervised Outlier Detection, March 2021. arXiv:2103.12051 [cs]. 21

[SG07] Raquel Sebastiao and Joao Gama. Change detection in learning histograms from data streams. In *Portuguese Conference on Artificial Intelligence*, pages 112–123. Springer, 2007. 69

[SGF+02] Ramasubramanian Sekar, Ajay Gupta, James Frullo, Tushar Shanbhag, Abhishek Tiwari, Henglin Yang, and Sheng Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 265–274, 2002. 5

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017. 80, 89, 98, 100, 101

[Sha53] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 1953. 15, 40, 43

[SHJ+17] Harini Suresh, Nathan Hunt, Alistair Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding using deep networks. *arXiv preprint arXiv:1705.08498*, 2017. 100, 101, 103

[Sip20] John Sipple. Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of device failure. In *International Conference on Machine Learning*, pages 9016–9025. PMLR, 2020. 62, 64

[SJLM23] Ashish Singh, Michael J Jones, and Erik G Learned-Miller. Eval: Explainable video anomaly localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18717–18726, 2023. 10

[SK10] Erik Strumbelj and Igor Kononenko. An Efficient Explanation of Individual Classifications Using Game Theory. *J Mach Learn Res*, 2010. 100

[SK23] Udo Schlegel and Daniel A Keim. A deep dive into perturbations as evaluation technique for time series xai. In *World Conference on Explainable Artificial Intelligence*, pages 165–180. Springer, 2023. 82

[SLY+21] Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and Evaluating Representations for Deep One-class Classification, March 2021. arXiv:2011.02578 [cs]. 21, 23

[SLYK20] Hyeok-Ki Shin, Woomyo Lee, Jeong-Han Yun, and HyoungChun Kim. {HAI} 1.0:{HIL-based} augmented {ICS} security dataset. In *13Th USENIX workshop on cyber security experimentation and test (CSET 20)*, 2020. 5

[SMK+21] Rohit Saluja, Avleen Kaur Malhi, Samanta Knapic, Kary Främling, and Cicek Cavdar. Towards a Rigorous Evaluation of Explainability for Multivariate Time Series. Technical report, arXiv, 2021. 41

[SMP+20] Min Sun, Stefano Moretti, Kelley Paskov, Nate Stockham, Maya Varma, Brianna Chrisman, Peter Washington, Jae-Yoon Jung, and Dennis Wall. Game theoretic centrality: a novel approach to prioritize disease candidate genes by combining biological networks with the Shapley value. *BMC Bioinformatics*, 2020. 41

[SOKEA20] Udo Schlegel, Daniela Oelke, Daniel A Keim, and Mennatallah El-Assady. An empirical study of explainable ai techniques on deep learning models for time series tasks. *arXiv preprint arXiv:2012.04344*, 2020. 82

[Spe61] Charles Spearman. The proof and measurement of association between two things. 1961. 38

[SQK+22] Tim Schneider, Chen Qiu, Marius Kloft, Decky Aspandi Latif, Steffen Staab, Stephan Mandt, and Maja Rudolph. Detecting Anomalies within Time Series using Local Neural Transformations, February 2022. arXiv:2202.03944 [cs]. 21, 23

[SR18] Mahsa Salehi and Lida Rashidi. A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]. *ACM SIGKDD Explorations Newsletter*, 20(1):13–23, 2018. 5

[SRTS18] Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 101

[SSFE23] Laya Rafiee Sevyeri, Ivaxi Sheth, Farhood Farahnak, and Shirin Abbasinejad Enger. Transparent anomaly detection via concept-based explanations. *arXiv preprint arXiv:2310.10702*, 2023. 10

[SSW+17] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017. 22

[STK+17] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 101

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017. 64, 80, 89, 100, 101, 103

[SW22] Tom Shenkar and Lior Wolf. ANOMALY DETECTION FOR TABULAR DATA WITH INTERNAL CONTRASTIVE LEARNING. page 26, 2022. 21, 23

[SWS+99] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999. 7

[SWSST00] Bernhard Scholkopf, Robert Williamson, Alex Smola, and John Shawe-Taylor. Sv estimation of a distribution's support. *Advances in Neural Information Processing Systems*, 41, 01 2000. 20, 22, 30

[SY14] 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*, pages 4–11, 2014. 7, 20

[SZ18] Manya Ali Salitin and Ali Hussein Zolait. The role of user entity behavior analytics to detect network attacks in real time. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–5. IEEE, 2018. 5

[SZK14] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 542–550. SIAM, 2014. 6

[SZN+19] 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. 11, 22, 71, 72, 116, 119, 120

[Tak19] Naoya Takeishi. Shapley Values of Reconstruction Errors of PCA for Explaining Anomaly Detection. In *ICDMW*, 2019. 41

[TCD+22] Sarthak Manas Tripathy, Ashish Chouhan, Marcel Dix, Arzam Kotriwala, Benjamin Klöpper, and Ajinkya Prabhune. Explaining anomalies in industrial multivariate time-series data with the help of explainable ai. In *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 226–233. IEEE, 2022. 10

[TCJ22] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022. 9, 20, 22

[TD04] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54:45–66, 2004. 7, 22

[TJ03] Marina Thottan and Chuanyi Ji. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204, 2003. 5

[TJC+20] Sana Tonekaboni, Shalmali Joshi, Kieran Campbell, David K Duvenaud, and Anna Goldenberg. What went wrong and when? Instance-wise feature importance for time-series black-box models. In *Advances in Neural Information Processing Systems*, volume 33, pages 799–809. Curran Associates, Inc., 2020. 83, 89

[TKWB21] Markus Thill, Wolfgang Konen, Hao Wang, and Thomas Bäck. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, 112:107751, 2021. 22

[TMJS20] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances, October 2020. arXiv:2007.08176 [cs, stat]. 21, 23

[TTL11] Swee Tan, Kai Ting, and Fei Tony Liu. Fast anomaly detection for streaming data. pages 1511–1516, 01 2011. 9

[UD09] Eugen Ursu and Pierre Duchesne. On modelling and diagnostic checking of vector periodic autoregressive time series models. *Journal of Time Series Analysis*, 30(1):70–96, 2009. 115

[vCHHL18] Tjeerd van Campen, Herbert Hamers, Bart Husslage, and Roy Lindelauf. A new approximation method for the Shapley value applied to the WTC 9/11 terrorist attack. *Soc. Netw. Anal. Min.*, 2018. 47

[VCM+18] Phongtharin Vinayavekhin, Subhajit Chaudhury, Asim Munawar, Don Joven Agravante, Giovanni De Magistris, Daiki Kimura, and Ryuki Tachibana. Focusing on what is relevant: Time-series learning and understanding using attention. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2624–2629. IEEE, 2018. 83

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 100, 103

[WBR+20] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive Training for Improved Out-of-Distribution Detection, July 2020. arXiv:2007.05566 [cs, stat]. 21, 23

[WC06] Tzu-Chiang Wu and Arbee LP Chen. Maintaining moving sums over data streams. In *International Conference on Advanced Data Mining and Applications*, pages 1077–1084. Springer, 2006. 12

[WK21] Renjie Wu and Eamonn Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 11

[WLM+22] Rui Wang, Chongwei Liu, Xudong Mou, Kai Gao, Xiaohui Guo, Pin Liu, Tianyu Wo, and Xudong Liu. Deep Contrastive One-Class Time Series Anomaly Detection, October 2022. arXiv:2207.01472 [cs]. 21, 23, 24

[WSM15] Bing Wang, Jie Sun, and Adilson E Motter. Detecting structural breaks in seasonal time series by regularized optimization. *arXiv preprint arXiv:1505.04305*, 2015. 115

[WXYL18] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018. 132

[XCZ+18] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference*, pages 187–196, 2018. 113

[XGT+23] Chunjing Xiao, Zehua Gou, Wenxin Tai, Kunpeng Zhang, and Fan Zhou. Imputation-based time-series anomaly detection with conditional weight-incremental diffusion models. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2742–2751, 2023. 9

[XRY+15] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015. 7

[XWWL21] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021. 9, 20, 23, 31

[YK20] Selim F Yilmaz and Suleyman S Kozat. Pysad: A streaming anomaly detection framework in python. *arXiv preprint arXiv:2009.02572*, 2020. 11

[YLLL22] Susik Yoon, Youngjun Lee, Jae-Gil Lee, and Byung Suk Lee. Adaptive model pooling for online deep anomaly detection from a complex evolving data stream. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2347–2357, 2022. 9

[YWP18] Shujian Yu, Xiaoyang Wang, and José C Príncipe. Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels. *arXiv preprint arXiv:1806.10131*, 2018. 46

[YWZ06] Ying Yang, Xindong Wu, and Xingquan Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery*, 13:261–289, 2006. 131

[YZZ+23] Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. *arXiv preprint arXiv:2306.10347*, 2023. 9

[ZCLZ16] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. In *International Conference on Machine Learning*, pages 1100–1109. PMLR, 2016. 67, 113

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014. 82, 89, 101

[ZFL+18] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018. 9

[ZJP+21] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021. 20

[ZK20] Di Zhao and Yun Sing Koh. Feature drift detection in evolving data streams. In *DEXA*, 2020. 39, 41

[ZLH+19] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *IJCAI*, volume 2019, pages 4433–4439, 2019. 9

[ZLW+21] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. *arXiv preprint arXiv:2112.00911*, 2021. 112, 114

[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019. 11

[ZPT+23] Qihang Zhou, Guansong Pang, Yu Tian, Shibo He, and Jiming Chen. Anomalyclip: Object-agnostic prompt learning for zero-shot anomaly detection. *arXiv preprint arXiv:2310.18961*, 2023. 9

[ZSM+18] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018. 22, 30, 62, 64, 67, 113

[ZSS+19] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. " why should you trust my explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019. 100

[ZSZ+21] Yingjie Zhou, Xucheng Song, Yanru Zhang, Fanxing Liu, Ce Zhu, and Lingqiao Liu. Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 22

[ZvdZP+19] Shihao Zheng, Simon B van der Zon, Mykola Pechenizkiy, Cassio P de Campos, Werner van Ipenburg, Hennie de Harder, and Rabobank Nederland. Labelless concept drift detection and explanation. In *NeurIPS Workshop on Robust AI in Financial Services*, 2019. 39, 41