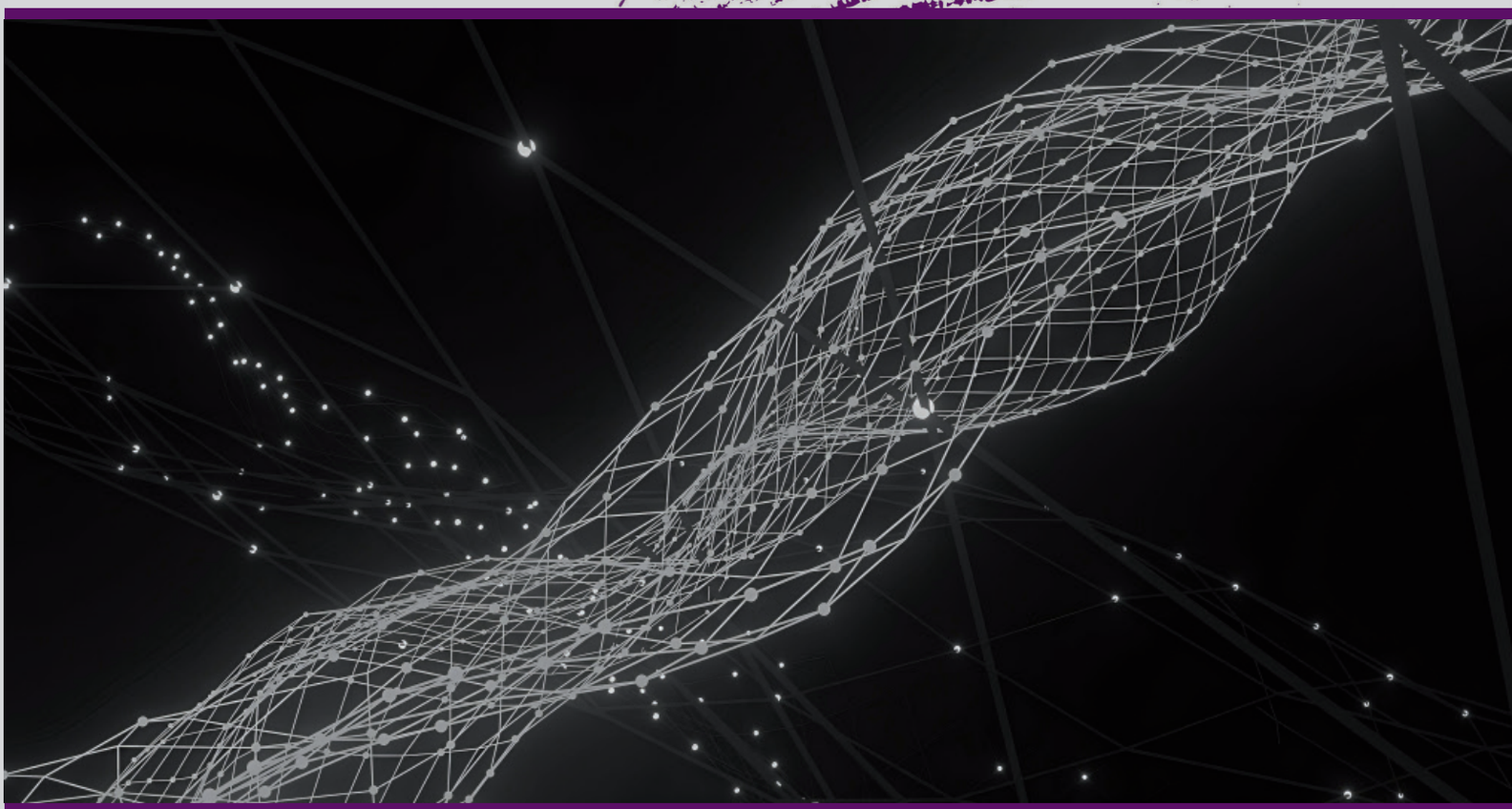
Mondragon Unibertsitatea

DOCTORAL THESIS

EXPLAINABLE ARTIFICIAL INTELLIGENCE FOR ANOMALY DIAGNOSIS IN MULTI-SENSOR DATA



JOKIN LABAIEN SOTO | Arrasate-Mondragón, 2023

JOKIN LABAIEN SOTO | Explainable Artificial Intelligence for Anomaly Diagnosis in Multi-sensor Data

Explainable Artificial Intelligence for Industrial Anomaly Diagnosis in Multi-Sensor Data

by

Jokin Labaien Soto

Supervised by Xabier De Carlos & Ekhi Zugasti

September, 2023

Acknowledgments

Despite the challenging nature of this thesis journey, it has been a path rich in learning, both technically and personally. This process has brought with it difficult moments, times when giving up seemed like an option, yet it has also provided moments of joy and satisfaction. Throughout the thesis, thankfully I've been surrounded by wonderful people who have made this journey more enjoyable. To all of these people, eskerrik asko!

Firstly, I want to express my gratitude to Ikerlan for providing me the opportunity to carry out my thesis here and for all the support they have given me to make it happen. I especially want to thank my teammates, who are part of those individuals who have eased this journey. I have truly enjoyed my daily interactions with them, both on a professional and personal level.

I would also like to give special thanks to my advisors, Xabier De Carlos and Ekhi Zugasti, for guiding me through this process and for providing technical help at moments when I wasn't sure which direction to take. It's been a pleasure to share this journey with both of you!

I can't forget Pin Yu Chen and Ide San, my advisors during my time at IBM Research. I learned a lot from both of you during this stay, and our discussions were always incredibly enriching. Thank you very much for the opportunity to work with you!

Finally, I want to thank my family and friends, who, after all, are the ones there in my day-to-day life and with whom I share most of my time. Eskerrik asko!

Zahartzea ez da okerragoa, ez bizitzea baino.
Fiachras.

Abstract

This thesis explores the potential of Explainable Artificial Intelligence (XAI) in the context of time-series anomaly detection and diagnosis. By enhancing the transparency of traditionally opaque models and offering immediate and intelligible explanations, we have set the stage for more informed decision-making processes across diverse sectors.

Prior to our practical exploration, we reviewed the existing literature on XAI, anomaly detection, and diagnosis. Then, our experiments start with a study of the Counterfactual Explanation Method (CEM) in time-series tasks. This investigation revealed both the advantages and limitations of CEM. Acknowledging its shortcomings, particularly its time-consuming nature. Then, we present the Real-Time Guided Counterfactual Explanations (RTGCEx) method. This innovative method is a model-agnostic approach that provides user-driven counterfactual explanations in real-time across different domains and data types.

Afterward, to avoid losing essential information that anomaly detectors might contain and that model-agnostic methods might miss, we address the challenge of creating intrinsically interpretable models. To achieve this, we first introduce the Diagnostic Fourier-based Spatio-temporal Transformer (DFS-Trans). This tool combines the capabilities of 1D Convolutional Neural Networks with a Transformer-inspired structure. This model effectively learns spatial and temporal dependencies in multivariate sensor data, proving to be a potent tool for diagnosing anomalies. Recognizing the challenges associated with obtaining labeled data, we developed an unsupervised variant, termed uDFSTrans. This model incorporates a dual strategy: a multi-masking technique and a context-oriented attention mechanism, facilitating the detection and elucidation of anomalies without the necessity for labeled data.

Laburpena

Tesi honek Explainable Artificial Intelligence-ek (XAI) denborazko serieetan anomaliak detektatzeko eta diagnostikatzeko duen ahalmena aztertzen du, Tesiak tradizionalki opakoak izan diren ereduen gardentasuna handitzean eta berehalako azalpen ulergarriak ematean hainbat sektoretan erabakiak hartzeko prozesu informatuagoak egiteko oinarriak ezartzen ditu.

Gure ahalegin esperimentalen aurretik, XAIri, anomalien detekzioari eta diagnostikoari buruz argitaratutakoa berrikusten da. Miaketa praktikoa denborazko serieen datuetan zentratutako Contrastive Explanation Method (CEM) algoritmoaren azterketarekin hasten da. Ikerketa honek CEM-en abantailak eta mugak erakusten ditu, eta bere gabeziak ezagutzean, zeinetatik geldotasuna nabarmentzen den, Real-time Guided Counterfactual Explanations (RTGCEx) izeneko metodoa aurkezten da. Eredu berritzaile hori azaldu beharreko modeloarekiko agnostikoa da eta erabiltzaileak gidatutako azalpenak denbora errealean ematea ahalbideratzen du hainbat domeinu eta datu motatarako.

Gure xedea anomalia detekziorako ereduen erabakiak zertan oinarritzen diren azaleratzea denez eta modelo agnostikoek barne funtzionamenduari erreparatzen ez diotenez, berez interpretagarriak diren eruduak sortzeko erronkari heltzen zaio informazio garrantzitsua gal ez dadin. Behar horri erantzuteko, lehenik eta behin, Diagnostic Fourier-based Spatio-temporal Transformer (DFSTrans) algoritmoa proposatzen da. Tresna honek dimentsio bateko sare neuronal konboluzionalen gaitasunak eta Transformerretan oinarritutako egitura bat konbinatzen ditu. Eredu honek aldagai anitzeko sentsoreen datuen mendekotasun espazialak eta tenporalak eraginkortasunez ikasten ditu anomaliak diagnostikatzeko tresna indartsua dela erakutsiz. Etiketatutako datuak lortzeko zailtasunak medio, uDFSTrans izeneko aldaera ez superbisatu bat garatzen dugu; izan ere, eredu honek, modu ez superbisatuan anomaliak antzeman eta diagnostikatzeko estrategia bikoitza erabiltzen du: maskaratze-teknika bat eta testuingurura bideratutako arreta-mekanismo bat.

ERRATUM LIST

Page	Reads now	Should be
9	Fig 1.6, "Example a"	Fig 1.6, "Example of a"
12	"advance. obtaining"	"Obtaining"
18	"comprehending"	"understanding"
19	Sec 2.1, "Explainable AI"	"XAI"
20	"between different classes"	"between different clusters"
26	"called SP-LIME"	(repeated)
27	"they define some"	"it defines some"
28	z=1 indicates...	z=0 indicates...
30	"as proximate as possible"	"as close as possible"
31	"class Let"	"class. Let"
31	"(Equation (2.7):"	"(2.7)):"
31	"The third term"	"The fourth term"
32	"change For"	"change. For"
34	"permutations"	"perturbations"
35	Fig 2.6, "Gaussian Mixture Models"	(repeated)
37	"relationships, so"	"relationships. So"
38	"biased"	"bias"
38	"mapping the inputs"	"mapping of the inputs"
38	"predictions algorithms"	"prediction algorithms"
39	Sun et al. [?]	(missing reference)
45	"optimizing it"	"optimizing them"
49	IF	iForest
57	"explainable artificial intelligence methods"	"XAI methods"
65	definicion de acrónimo CEM repetida	Should be written CEM directly.
70	"proved"	"tested"
72	Figs 4.7, 4.8 y 4.9 deberían ser subfigs de una misma figura	Figs 4.7, 4.8 y 4.9 deberían ser subfigs de una misma figura
77	f(x) (Eq. 5.4)	Debería de ser G(x)
82	"is input"	"is taken as input"
86	"dataset."	"datasets."
90	Figs 5.10 y 5.11 deberían ser subfigs de 5.12	Figs 5.10 y 5.11 deberían ser subfigs de 5.12
101	Eq (6.1), left square bracket should be round bracket	Eq (6.1), left square bracket should be round bracket
102	"matrxices"	"matrices"
102	"that varies "	which varies
103	footnote 1 printed on page 104	Should be printed on page 103
104	Sec 6.3, "problem setting"	"overall architecture"

Page	Reads now	Should be
114	"these state-of-the-art"	"the state-of-the-art"
116	"defined in Section A"	defined in Section 6.3.3"
133	Commas missing in the enumeration of the algorithms	Enumeration of algorithms using commas where needed.
133	"Section 2.1 We"	"Section 2.1. We"
134	"heartbeats Type"	"heartbeats. Type"
135	"protocolm"	"protocol"
136	Sec 7.5.1: "in Table ??"	Sec 7.5.1: "in Table 7.1"
138	Fig 7.5 (overlapping text in vertical axis)	Fig 7.5 (should be without overlapping text in the vertical axis)
145	"diagnosis.."	"diagnosis."
148	"black box models"	"black-box models"
153	"This paper studies"	"This thesis studies"

Resumen

Esta tesis explora el potencial de la Inteligencia Artificial Explicable (XAI) en el contexto de la detección y diagnóstico de anomalías en series temporales. Al aumentar la transparencia de modelos tradicionalmente opacos y ofrecer explicaciones inmediatas e inteligibles, hemos sentado las bases para procesos de toma de decisiones más informados en diversos sectores.

Antes de empezar con la fase experimental, revisamos la literatura existente sobre XAI, detección de anomalías y diagnóstico de anomalias. La exploración práctica comienza con un estudio del algoritmo Contrastive Explanation Method (CEM) en tareas de series temporales. Esta investigación revela tanto las ventajas como las limitaciones del CEM. Tras reconocer sus deficiencias, en particular su lentitud, presentamos el método llamado Real-time Guided Counterfactual Explanations (RTGCEx). Este innovador método es un enfoque agnóstico del modelo que proporciona explicaciones contrafácticas guiadas por el usuario en tiempo real para diferentes dominios y tipos de datos.

Posteriormente, para evitar la pérdida de información esencial que podrían contener los detectores de anomalías y que los métodos agnosticos podrían pasar, abordamos el reto de crear modelos intrínsecamente interpretables. Para ello, presentamos en primer lugar el algoritmo Diagnostic Fourier-based Spatio-temporal Transformer (DFSTrans). Esta herramienta combina las capacidades de las redes neuronales convolucionales 1D con una estructura inspirada en los Transformers. Este modelo aprende eficazmente las dependencias espaciales y temporales de los datos temprales multivariantes, demostrando ser una potente herramienta para diagnosticar anomalías. Reconociendo las dificultades asociadas a la obtención de datos etiquetados, desarrollamos una variante no supervisada, denominada uDFSTrans. Este modelo incorpora una doble estrategia: una técnica de multienmascaramiento y un mecanismo de atención orientado al contexto, que facilita la detección y elucidación de anomalías sin necesidad de datos etiquetados.

Contents

List of Figures

List of Tables

1

Introduction

This dissertation addresses various challenges in Explainable Artificial Intelligence (XAI) as applied to the field of anomaly detection, with a primary focus on enhancing the diagnosis of anomalies. In order to provide a comprehensive understanding of the research presented, the current chapter offers essential background information. To begin, Section 2.1 presents an overview of XAI, including fundamental definitions and a taxonomy of XAI methods. Following this, Section 1.2 introduces key concepts related to anomaly detection and diagnosis, as well as a taxonomy for categorizing different approaches. Section 1.3 then explains the motivation behind this thesis, and Section 3.1 outlines the hypothesis and objectives of the research. Lastly, Section 1.4 provides an overview of the dissertation structure.

1.1 Explainable Artificial Intelligence

Artificial Intelligence (AI) and Machine Learning (ML) have made remarkable strides in recent years, giving rise to sophisticated models that excel at various tasks across domains such as healthcare, finance, and natural language processing. Deep neural networks exemplify such models, but their increasing complexity has led to a lack of interpretability, commonly known as the *black-box* problem. This absence of interpretability presents challenges for human users who need to comprehend, trust, and validate the models' decisions for ethical, legal, and practical reasons.

Explainable Artificial Intelligence (XAI) is a subfield of AI dedicated to making the decision-making process of ML models more comprehensible and interpretable for humans. The growing complexity of AI models, including deep neural networks, has caused the *black-box* problem to become more pronounced. XAI tackles this issue by offering insights into the rationale behind a model's predictions, allowing users to trust and validate the decisions made by the model.

1.1.1 Concepts and definitions

The most common concepts related to XAI will be defined in this section. Defining these concepts is important for providing a foundation for better understanding the following sections on XAI. Before delving into these definitions, we define the Explainable Artificial Intelligence concept:

Definition 1. *(Explainable AI). Explainable AI refers to the effort of making the decision-making process of machine learning models more understandable and interpretable for humans.*

A. Model Understandability and Accessibility

The transparency and understandability of machine learning models are central to their acceptance and usability. The concepts of *black-box* and *white-box* models provide insights into this spectrum.

Definition 2. *(Black-box). Black-box models refer to machine learning models whose decision-making process is opaque and incomprehensible to humans.*

Definition 3. *(White-box). White-box models refer to machine learning models whose decision-making process is transparent and understandable to humans.*

These terms relate to the three main concepts of XAI: *explainability*, *interpretability*, and *transparency*.

Definition 4. *(Explainability). Explainability refers to the provision of knowledge about the internal functioning of an algorithm and clarifying **why** it behaves in a certain way in human terms.*

Definition 5. *(Interpretability). Interpretability refers to the ability to understand and explain the relationship between the algorithm's input, output, and internal workings. Interpretability clarifies **how** an algorithm reaches a decision.*

Definition 6. *(Transparency). The transparency of a model refers to the degree of comprehensibility and interpretability of a specific model by itself, that is, whether the overall functioning of the model, its individual components, and its learning algorithm are intelligible or understandable to a human.*

—

B. Model Ethics and Integrity

Using machine learning models responsibly and understanding them properly is crucial. Key ideas include *algorithmic bias*, *robustness*, *fairness*, and *trustworthiness*.

Definition 7. *(Algorithmic bias). Algorithmic bias refers to the systematic and repeated errors or unfair outcomes that can result from using AI systems, usually when the data used to develop and train the algorithm contains implicit or explicit biases.*

Definition 8. *(Robustness). Robustness in machine learning refers to the capability of a model to maintain its performance level despite the presence of errors, perturbations, or variations in the input or execution environment.*

Definition 9. *(Fairness). Fairness refers to the absence of unjustified or discriminatory biases towards individuals or groups based on their protected characteristics, such as gender, race, or age, in the development, deployment, and use of machine learning models.*

All these considerations inform a model's *trustworthiness*.

Definition 10. *(Trustworthiness). Trustworthiness refers to the extent to which users can rely on the predictions, decisions, or recommendations generated by machine learning models.*

—

C. Explanation Techniques

To better understand and trust machine learning models, various techniques and explanations have been developed, including *feature importance*, *counterfactual explanation*, *rule-based explanation*, *gradient-based attribution*, and *attention-based explanation*.

Definition 11. *(Feature importance). Explanations based on feature importance are techniques used to measure the significance or impact of each input or feature on the model's output or decision.*

Definition 12. *(Counterfactual explanation). Counterfactual explanations describe how a model's output or decision would have changed if certain input variables or features had been different.*

Definition 13. *(Rule-based explanation). Rule-based explanation refers to explanations that are based on explicit rules or decision-making criteria used by a model to arrive at a prediction or decision.*

Definition 14. *(Gradient-based attribution). Gradient-based attribution methods give insights into a model's decision by showing the gradient of the output with respect to the input.*

Definition 15. *(Attention-based explanation). Attention-based explanations provide insights by highlighting parts of the input that the model finds most relevant when making a prediction. It can help visualize which parts of the input data the model is focusing on.*

1.1.2 Taxonomy

To systematically study and apply XAI methods, it is essential to classify them based on their characteristics and goals. The literature offers numerous taxonomies to distinguish between these methods. For the purposes of this thesis, we adopt a taxonomy akin to that presented in Kamth et al.'s book [1]. As shown in Figure 1.1 this categorization system delineates explainability techniques into three overarching groups: interpretability scope, application stage, and model depedency.

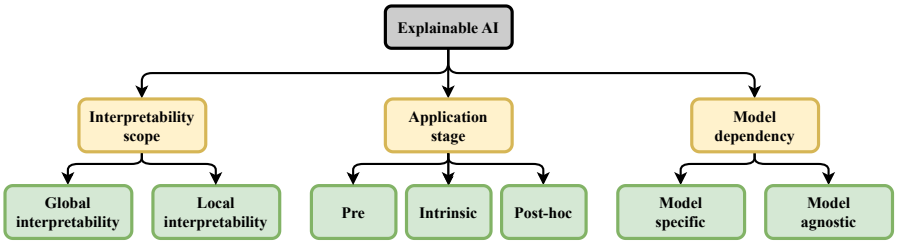


Fig. 1.1: Taxonomy of Explainable AI.

A. Interpretability scope.

Interpretability methods can be classified based on the scope of interpretability, which encompasses two primary categories: global interpretability and local interpretability. **Global interpretability** aims to understand a model's functionality across the entire dataset, while **local interpretability** focuses on explanations for specific instances within the dataset. The difference between both interpretability scopes is illustrated in Figure 1.2.

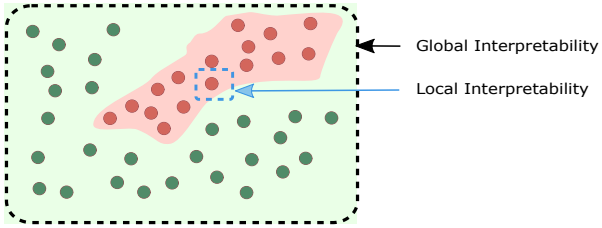


Fig. 1.2: Global vs local interpretability.

Methods for interpretability are different depending on the scope:

1. **Local methods:** These explainers offer localized explanations by elucidating a model's output for a specific instance. Typically, such explanations are generated by assessing the contribution of individual features for a particular prediction derived from a given input or by approximating the model within a small region of interest using a simpler model.
2. **Global methods:** Employing the entire training dataset, global explainers provide insights into the overall influence of input features on the model's predictions. Consequently, these explanations enable a deeper understanding of how the model's structures and parameters impact its predictions, ultimately fostering greater transparency in the decision-making process.

B. Application stage.

Interpretability models can also be classified based on their application stage. Depending on when interpretability methods are applied, they can be divided into three groups: (i) pre-model explanations, which are applied to the data before training the models; (ii) intrinsic explanations, which are obtained using model internals; (iii) and post-hoc explanations, which are applied after training the model without relying on model internals.

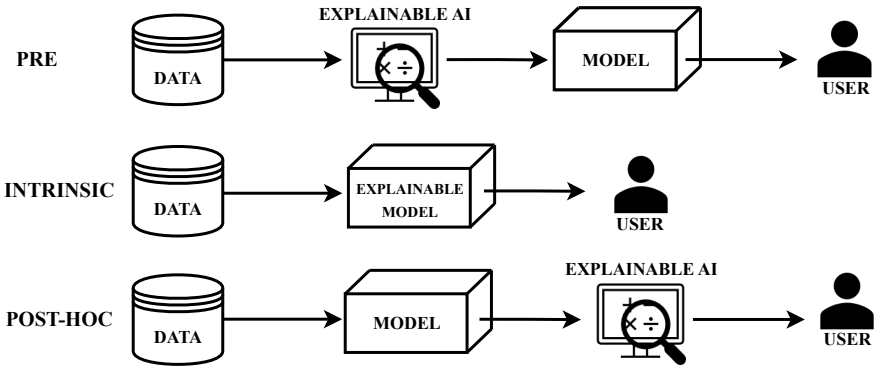


Fig. 1.3: Explainable AI methods classified by the application stage.

Figure 1.3 shows the differences between the three groups and they are described below:

1. **Pre-model** interpretability, as the name suggests, is applied before training the model, on the input data. This interpretability is used to understand and process the data and involves exploratory data analysis, knowledge graphs, feature engineering, or data visualization. The pre-model interpretability is essential for the design of the model.

2. **Intrinsic** interpretability refers to the ability of an algorithm or model to provide an explanation of its decision-making process inherently within the algorithm itself, without requiring any additional tools.
3. **Post-hoc** interpretability. While intrinsic interpretability is a built-in explanation within the algorithm, post-hoc methods are techniques or tools applied after an algorithm or model has made its decisions to understand or explain its behavior. Post-hoc methods can be applied either to intrinsically interpretable methods or to opaque black-box models.

C. Model dependency.

Model dependency serves as an alternative classification scheme for XAI techniques, indicating if an explainability approach is universally applicable to all models (model-agnostic) or designed exclusively for certain model types (model-specific).

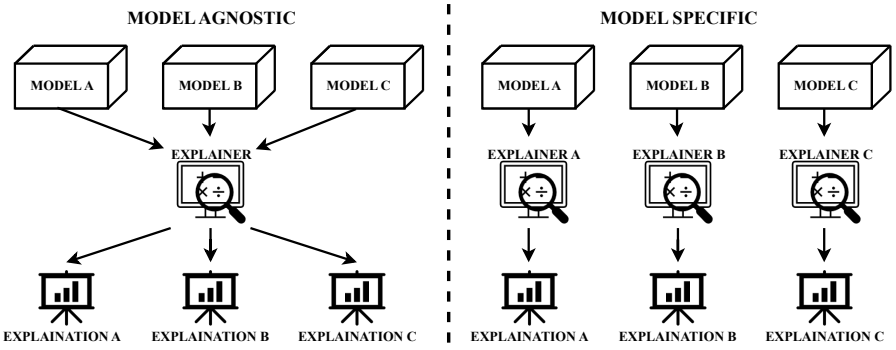


Fig. 1.4: Explainable AI methods classified by model dependency.

The differences between these two groups are illustrated in Figure 1.4 and described below:

1. **Model-specific methods:** These techniques can only be applied to specific models, as they are designed to exploit the internal structure of the model to generate explanations. Generally, these methods yield more detailed explanations compared to model-agnostic approaches. For instance, attention mechanisms fall into this category.
2. **Model-agnostic methods:** These techniques require only the input and output of a model, without needing access to the model's internal structure, allowing their application across various models. Techniques such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) serve as examples of model-agnostic approaches.

1.2 Anomaly detection and diagnosis

The **detection of anomalies** in time-series consists of identifying unusual or anomalous patterns or events that significantly differ from the normal or expected behavior of a data series over time. This involves analyzing and comparing the historical values and trends of the time-series with current data, in order to detect any significant deviation that may indicate an anomaly. On the other hand, the **diagnosis of anomalies** involves providing more information about the anomaly that has been detected at a certain moment, as well as reporting which sensors have been affected at that time. The combination of detection and diagnosis of anomalies in time-series is crucial for making informed decisions and preventing major damage in critical systems and processes. Both detection and diagnosis are important aspects in anomaly detection, where detection identifies when a deviation from the expected behavior occurs, while diagnosis seeks to identify the cause and impact of the anomaly and provide further information to aid in decision-making.

1.2.1 Concepts and definitions

This section defines some concepts related to AD in time-series data.

Definition 16. *(Time-series). A time-series* $\mathsf{X} = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_T\}$ *can be described as a sequence of ordered, real-valued observations of length* T*, where at each timestamp* t*,* $\boldsymbol{x}_t \in \mathbb{R}^m$ *represents a set of points collected from* m *different sensors. Similarly, we can define a corresponding label sequence* $\mathsf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$ *for the time-series* X*.*

A time-series can be categorized as either *univariate* or *multivariate* based on the number of sensors used to collect observations.

Definition 17. *(Univariate time-series). A univariate time-series is a time-series in which the observations are collected from a single sensor. That is,* $\mathsf{X} = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_T\}$*, where* $\boldsymbol{x}_t \in \mathbb{R}$

Definition 18. *(Multivariate time-series). A multivariate time-series is a time-series in which the observations are collected from multiple sensors. That is,* $\mathsf{X} = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_T\}$*, where* $\boldsymbol{x}_t \in \mathbb{R}^m$*, being* $m > 1$*.*

In this context, we can distinguish between two concepts: *anomaly detection* and *anomaly diagnosis*.

Definition 19. *(Anomaly detection). Anomaly detection refers to the process of predicting the label* $\boldsymbol{y}_t \in \{0, 1\}$ *for an unseen datapoint* $\boldsymbol{x}_t$*. A label of 0 indicates that the point* $\boldsymbol{x}_t$ *is normal, while a label of 1 indicates that it is anomalous.*

Definition 20. *(Anomaly diagnosis). Given an anomalous point $\boldsymbol{x}_t$, anomaly diagnosis is the process of providing additional information $\mathcal{I}$ about the detected anomaly. This information can be given in different ways:*

- ***Visualization of features**: Visual analysis of the features associated with the anomaly.*
- ***Information based on attention mechanisms**: Utilizing attention mechanisms to focus on specific data components that contribute to the anomaly.*
- ***Information based on feature importance**: Quantifying the significance of each feature in the anomaly detection process.*
- ***Sensor-wise label prediction:** Predicting the label $\boldsymbol{y}_t = \{0,1\}^m$ corresponding to each of the m sensors at timestamp t.*

Generally, the anomalies that can be found in AD problems are categorized into three groups: *point anomalies*, *subsequence anomalies*, and *time-series anomalies*. Each group is described next:

A. Point anomalies

Definition 21. *(Point anomaly). Point anomalies refer to individual points which are far outside the normality ranges of the time-series to which they belong.*

It is the simplest type of anomaly and most of the AD research has been focused on detecting these types of anomalies. Some point anomalies are illustrated in Figure 1.5

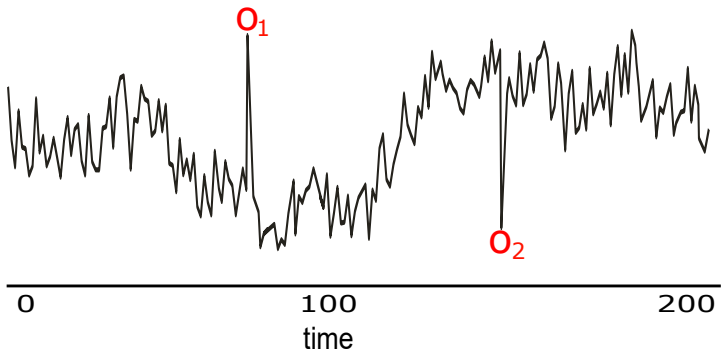


Fig. 1.5: Example of a univariate time-serie containing two point anomalies, highlighted in red.

B. Subsequence anomalies

A subsequence anomaly is a pattern or sequence of values that stand out as being significantly different from the rest of the data. Two types of anomalies can be distinguished within this category: *contextual anomalies* or *collective anomalies*.

Definition 22. *(Contextual anomaly). Contextual anomalies refer to a set of points that are anomalous in a specific context, but not otherwise.*

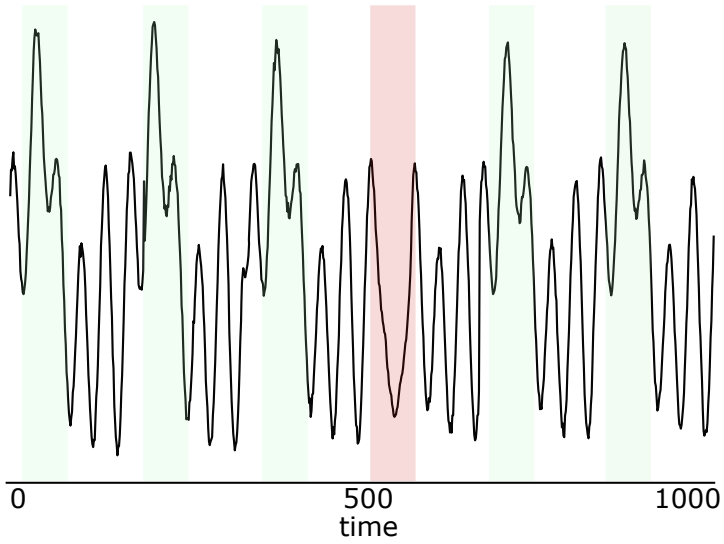


Fig. 1.6: Example a univariate time-series containing contextual anomalies, which are highlighted in red.

In Figure 1.6, we present a univariate time-series that exhibits a contextual anomaly, which is visually identified by highlighting it in red. The red highlighted points display low values instead of the expected high values, indicating that they deviate from the usual contextual behavior.

Definition 23. *(Collective anomalies). Collective anomalies refer to a collection of related data points that are anomalous with respect to the entire time-series. The instances that belong to that family, may not be singularly anomalous but the occurrence of all of them together is anomalous.*

An example of a collective anomaly is illustrated in Figure 1.7, where the occurrence of several points during the period highlighted in red does not correspond to the expected pattern of the data.

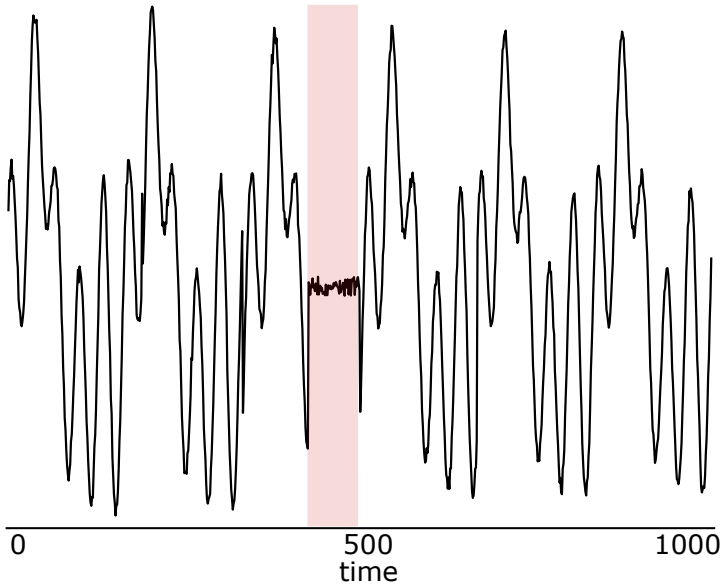


Fig. 1.7: Example of a collective anomaly.

C. Time-series anomaly

Definition 24. *(Time-series anomaly). A time-series anomaly is a time-series* X_i *that, as a whole, deviates significantly from the other time-series in a given dataset* $\mathcal{D} = \{\mathsf{X}_1, \ldots, \mathsf{X}_N\}$ *of univariate or multivariate time-series. This can be seen as a generalization of subsequence outliers to a set of time-series.*

An example of a time-series anomaly is illustrated in Figure 1.8, in which the red colored time-series is anomalous with respect the the other time-series-

1.2.2 Taxonomy

In order to systematically classify deep learning algorithms employed for anomaly detection in time-series data, we will adhere to the taxonomic framework illustrated in Figure 1.9. This framework organizes the algorithms using two primary criteria.

Initially, the algorithms are sorted according to their learning paradigms, which can be segregated into three distinct categories: supervised learning, semi-supervised learning, unsupervised learning, and self-supervised learning. Subsequently, the algorithms are further categorized based on the objective of their training objective, resulting in four different approaches: classification-based models, forecasting models, reconstruction-centric models, and clustering techniques.

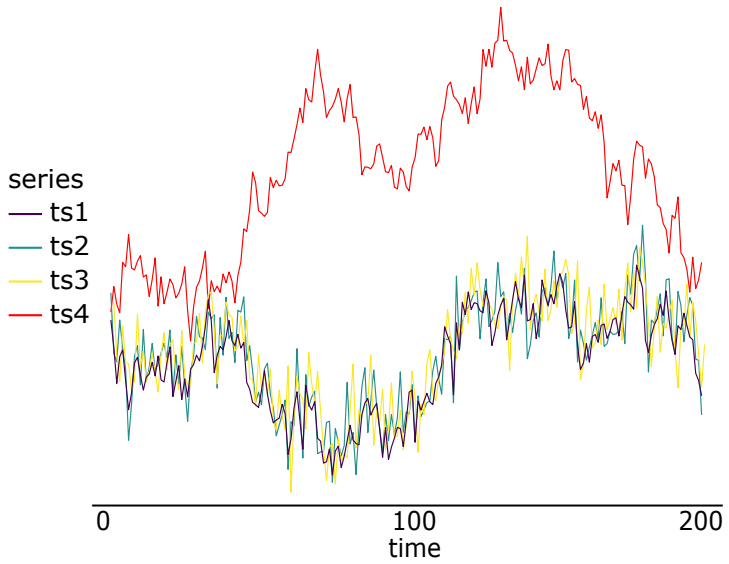


Fig. 1.8: Example of a time-series anomaly. The red colored time-series is anomalous with respect to the other time-series.

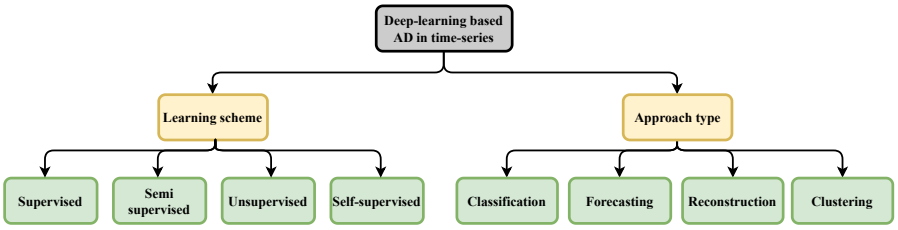


Fig. 1.9: Taxonomy of deep learning approaches for time-series anomaly detection.

1.2.2.1 Learning schemes

Which learning approach to use often depends on the type of data at hand and what the task demands. There are four main ways to learn: supervised, semi-supervised, unsupervised, and self-supervised. The following section explains each of these methods, their workings, benefits, and challenges.

A. Supervised learning

As depicted in Figure 1.10, in the supervised learning scheme, the model is trained on labeled data, using a training set with observations labeled as either normal or anomalous. The goal is to teach the model to distinguish

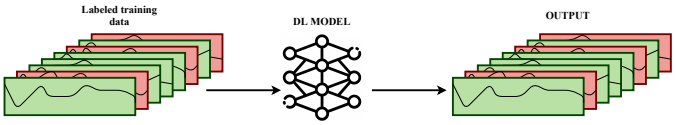


Fig. 1.10: Supervised learning scheme.

between normal and anomalous instances. Supervised learning can be particularly effective when a sufficient amount of labeled data is available for both normal and anomalous cases. However, given that anomalies tend to be infrequent or hard to characterize in advance. obtaining such labeled data can be challenging.

B. Semi-supervised learning

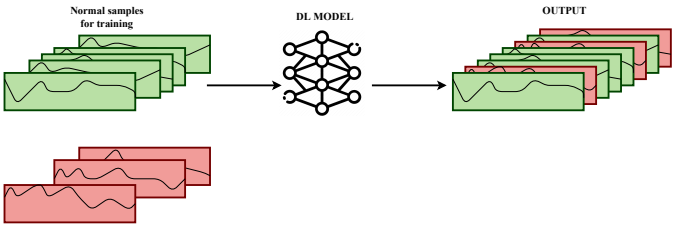


Fig. 1.11: Semi-supervised learning scheme.

When acquiring labeled anomalous data proves challenging, or when only samples of normal data are available, the semi-supervised learning scheme is employed. Models trained in a semi-supervised manner for anomaly detection utilize only normal samples during the training process. This approach enables the model to capture the essence of normality, subsequently facilitating the detection of atypical occurrences. Figure 1.11 illustrates the semi-supervised learning scheme.

C. Unsupervised learning

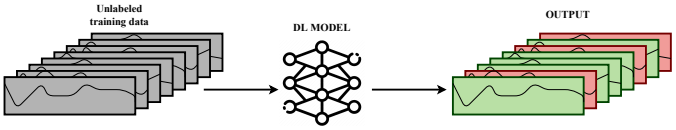


Fig. 1.12: Unsupervised learning scheme.

In the unsupervised learning scheme, no label information is utilized, neither explicitly nor implicitly, during the training process. The models are designed to autonomously discover underlying patterns or features that facilitate the distinction between normal and anomalous instances. This approach is particularly advantageous when labeled data is scarce or not readily obtainable, as the models can learn to identify inherent structures or relationships in the data without any prior knowledge. Figure 1.12 illustrates how unsupervised models are trained.

D. Self-supervised learning

In the self-supervised learning scheme, label information may or may not be used directly. The key idea is to leverage a re-labeling strategy or auxiliary task to create surrogate labels from the data itself. These surrogate labels are then used to train the model in a supervised manner, even though the original labels may not be available or utilized. Figure 1.13 illustrates how self-supervised approaches work.

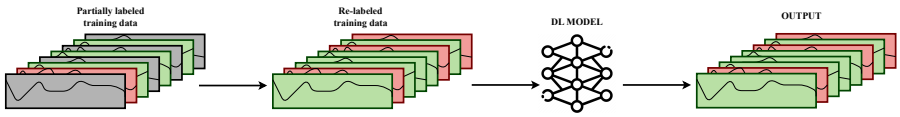


Fig. 1.13: Self-supervised learning scheme.

1.2.2.2 Training objective

When discussing anomaly detection, it is crucial to understand the varied training objectives, as each offers a unique approach to identifying irregularities within data. In this section, we distinguish between four primary strategies: classification, forecasting, reconstruction, and clustering.

A. Classification

Classification methods in anomaly detection involve supervised training of algorithms to categorize time-series data as either anomalous or non-anomalous. To train these models, a dataset $\mathcal{D} \triangleq \{(\mathsf{X}_i, y_i) \mid i = 1, \ldots, N\}$ consisting of labeled time-series is used, where each instance $\mathsf{X}_i \in \mathbb{R}^T$ represents a time-series of length T and $y_i \in \{0, 1\}$ denotes its corresponding binary label. Here, the label $y_i = 0$ indicates that the time-series X_i is non-anomalous, while $y_i = 1$ signifies the presence of an anomaly within the time-series or the overall time-series being considered anomalous. A model $\mathcal{M}$ is defined as a function $\mathcal{M} : \mathbb{R}^T \to \{0, 1\}$ that performs the mapping from the input time-series X_i to the binary labels y_i. An example of how a classification model works classifying anomalies is shown in Figure 1.14.

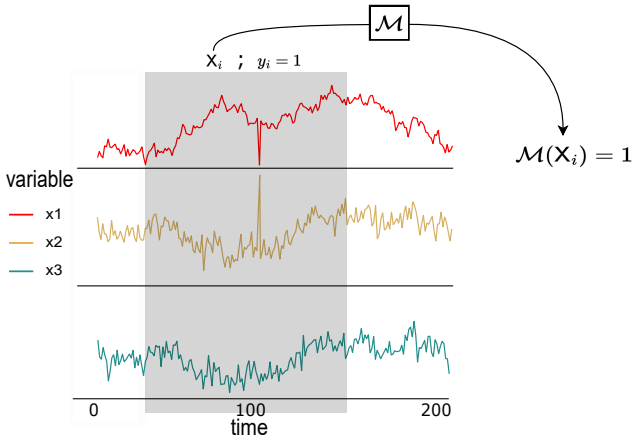


Fig. 1.14: Classification approach.

The error between the predicted labels $\mathcal{M}(\mathsf{X}_i)$ and the actual labels y_i can be measured using an objective function $\mathcal{L}(\mathcal{M}, \mathcal{D})$. One commonly used objective function in this context is the binary cross-entropy loss, defined as:

$$\mathcal{L}(\mathcal{M}, \mathcal{D}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\mathcal{M}(\mathsf{X}_i)) + (1 - y_i)\log(1 - \mathcal{M}(\mathsf{X}_i))\right]. \tag{1.1}$$

Other objective functions or metrics can also be employed, depending on the specific requirements of the problem and the characteristics of the data.

B. Forecasting

Forecasting-based approaches in anomaly detection rely on predicting future values in a time-series and comparing them with the actual values to identify anomalies. Given a time-series $\mathsf{X} \in \mathbb{R}^T$ of length T, the dataset $\mathcal{D} \triangleq \{(\{\boldsymbol{x}_{t-1-K}, ..., \boldsymbol{x}_{t-1}\}, \{\boldsymbol{x}_t, ..., \boldsymbol{x}_{t+n-1}\}), \mid t = K+1, ..., T\}$ used for training these methods is usually obtained by employing a sliding window approach. That is, given a segment $\{\boldsymbol{x}_{t-1-K}, ..., \boldsymbol{x}_{t-1}\} \in \mathbb{R}^K$ of size K, the model is trained to predict the next $n \geq 1$ values in the sequence, as illustrated in Figure 1.15. The objective is to assess the discrepancy between the predicted values and the actual values, and based on a predefined threshold τ, determine whether the predicted points are anomalous or not. Unlike classification methods, labels are not explicitly used in this approach. Instead, the models are typically trained in a semi-supervised or unsupervised manner, as defined in the previous section.

A model $\mathcal{M}$ is defined as a function $\mathcal{M} : \mathbb{R}^K \rightarrow \mathbb{R}^n$, where K is the window size of the input time-series and n is the number of values to predict. Defining $\mathsf{X}_t = \{\boldsymbol{x}_{t-1-K}, ..., \boldsymbol{x}_{t-1}\}$ and $\mathsf{Y}_t = \{\boldsymbol{x}_t, ..., \boldsymbol{x}_{t+n-1}\}$, the objective

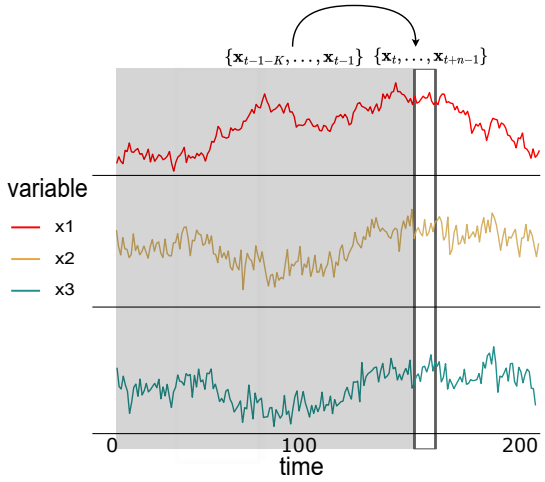


Fig. 1.15: Forecasting approach.

function $\mathcal{L}(\mathcal{M},\mathcal{D})$ measures the error between the predicted values $\mathcal{M}(\mathsf{X}_t)$ and the actual values Y_t. One common loss function used for training this type of model is the mean squared error, defined as:

$$\mathcal{L}(\mathcal{M},\mathcal{D}) = \frac{1}{N}\sum_{t=K+1}^{T}\left[\mathcal{M}(\mathsf{X}_t)-\mathsf{Y}_t\right]^2. \tag{1.2}$$

C. Reconstruction

Reconstruction-based approaches in anomaly detection involve reconstructing certain values (or all) in a time-series given a window of input values and comparing the reconstructed values with the actual values to identify anomalies. Given a time-series $\mathsf{X} \in \mathbb{R}^T$ of length T, the dataset $\mathcal{D} \triangleq \{(\{\boldsymbol{x}_{t-k_1},\ldots,\boldsymbol{x}_{t-k_2}\},\{\boldsymbol{x}_{t-n_1},\ldots,\boldsymbol{x}_{t+n_2}\}) \mid t = k_1,\ldots,T-k_2\}$ used for training these methods is usually obtained by employing a sliding window approach, where $0 \leq n_1 \leq k_1$ and $0 \leq n_2 \leq k_2$. That is, given a segment $\{\boldsymbol{x}_{t-k_1},\ldots,\boldsymbol{x}_{t-k_2}\} \in \mathbb{R}^K$ of size $K = k_1+k_2+1$, the model is trained to reconstruct the target values $\{\boldsymbol{x}_{t-n_1},\ldots,\boldsymbol{x}_{t+n_2}\} \in \mathbb{R}^n$, where $n = n_2+n_1+1$. The objective is to assess the discrepancy between the reconstructed values and the actual values, and based on a predefined threshold τ, determine whether the reconstructed points are anomalous or not. As in forecasting models, the model is trained semi-supervisedly or unsupervisedly. Figure 1.16 illustrates how reconstruction-based models work.

A reconstruction-based model $\mathcal{M}$ is defined as a function $\mathcal{M}:\mathbb{R}^K \to \mathbb{R}^n$, where K is the window size of the input time-series and n is the number of values to reconstruct. Defining $\mathsf{X}_t = \{\boldsymbol{x}_{t-k_1},\ldots,\boldsymbol{x}_{t+k_2}\}$ and $\mathsf{Y}_t =$

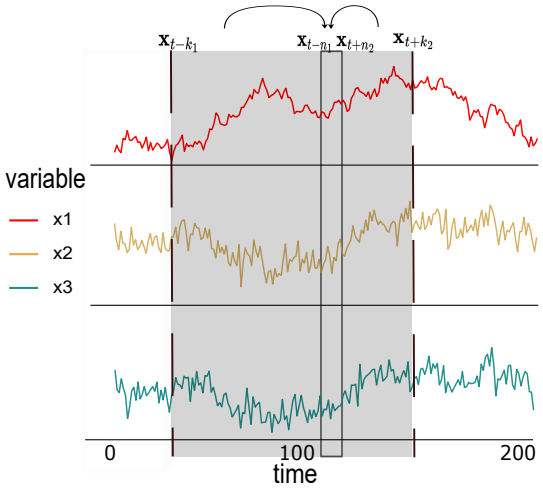


Fig. 1.16: Reconstruction-based approach.

$\{\boldsymbol{x}_{t-n_1}, \ldots, \boldsymbol{x}_{t+n_2}\}$, the objective function $\mathcal{L}(\mathcal{M}, \mathcal{D})$ measures the error between the reconstructed values $\mathcal{M}(\mathsf{X}_t)$ and the target values Y_t.

D. Clustering

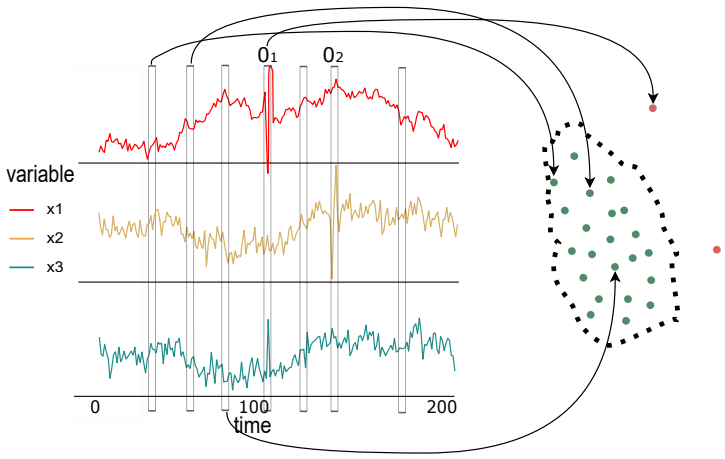


Fig. 1.17: Clustering approach.

Clustering-based approaches in anomaly detection involve grouping similar time-series data points together and identifying anomalies as data points that are distant from their corresponding cluster centroids. Given a time-series

$\mathsf{X} \in \mathbb{R}^T$ of length T, the dataset $\mathcal{D} \triangleq \{\boldsymbol{x}_t \mid t = 1, \ldots, T\}$ used for training these methods is usually composed by points or features extracted from the time-series. In clustering-based approaches, the aim is to identify data points that are distant from their corresponding cluster centroids or those that do not fit well into any of the clusters. Usually, these data-points are categorized into two clusters, one cluster representing normal data and the other containing data falling outside the normality. The data is often mapped into a lower-dimensional feature space and then a hyperplane is used to separate these two clusters, with the data points falling outside the normality considered anomalous, as shown in Figure 1.17. Similarity is often measured using distance metrics, such as Euclidean distance.

A clustering model $\mathcal{M}$ can be represented as a function $\mathcal{M} : \boldsymbol{x}_t \in \mathsf{X} \rightarrow \{1, \ldots, K\}$ that assigns each data point in the time-series to one of the K clusters. The clustering objective function $\mathcal{L}(\mathcal{M}, \mathcal{D})$ evaluates the quality of the clustering, often by measuring the sum of the distances between the data points and their corresponding cluster centroids. By minimizing this sum, the algorithm seeks to create compact and well-separated clusters.

1.3 Motivation

The motivation of this project is to address the needs of the industry for artificial intelligence (AI) solutions, specifically in the area of anomaly and defect detection in industrial processes. In such processes, even minor failures can result in significant financial losses for a company, highlighting the critical importance of detecting anomalies and defects. To achieve this, numerous sensors collect data on machine operation, which can be utilized to train Deep Learning (DL) models for anomaly detection. Despite the potential benefits of these models, **the data is often underutilized in industry**, resulting in a lag in comparison to academic research. However, **there is a growing trend in industry to adopt DL models for automatic anomaly detection**, and they have shown promising results. Nonetheless, the complexity of these models can hinder understanding of their predictions. Thus, there is a **need to balance the effectiveness of these models with their interpretability to increase confidence and trust in DL solutions in industry**. Key objectives for the development of Explainable AI (XAI) include trustworthiness, ethics, causality, and improbability, which should be prioritized in the pursuit of more interpretable and trustworthy AI solutions for the industry.

- **Trustworthiness:** AI systems often face greater scrutiny when making mistakes compared to humans, as humans can explain their decisions while AI systems usually cannot. Prioritizing interpretability helps reveal the decision-making process, fostering trust in the model.
- **Ethics:** In certain situations, AI models can significantly impact individuals' lives, such as in medicine, defense, justice, and autonomous vehicles.

Ethics play a crucial role in these cases, making XAI essential to prevent negative outcomes. The UK Parliament Select Committee on Artificial Intelligence has stated that deploying AI systems with substantial impact on an individual's life is unacceptable unless they can provide a full and satisfactory explanation for their decisions.

- **Causality:** Comprehending the underlying causal mechanisms of problems is vital in many domains. For instance, in industrial process anomaly detection, understanding the rationale behind a decision can help identify the root cause of an issue.
- **Improbability:** Gaining insights into the causes of decisions, the influence of variables on the final decision, and the model's inner workings can contribute to enhancing the model's performance.

1.4 Outline of the Dissertation

This thesis is organized into different sections. First, Chapter 2 covers the state of the art in the field of explainable artificial intelligence (XAI) and anomaly detection (AD) in time-series data. Based on the critical analysis of the state of the art, in Chapter 3 we present the hypothesis and the objectives that we set for filling the gaps found in the literature, and we present the contributions of the thesis. Following this, Part I delves into the contributions related to counterfactual explanations. In this part, Chapter 4 presents an application of a well-known CEM method for time-series data and Chapter 5 introduces a novel counterfactual explanation method for generating real-time counterfactual explanations in time-series data. Moreover, Part II focuses on the contributions related to transformer-based approaches for anomaly detection and diagnosis. Here, Chapter 6 presents a supervised spatio-temporal dependency discovery framework, which can provide readily consumable diagnostic information in both spatial and temporal directions in multi-sensor anomaly detection scenarios. To finish with the contributions, Chapter 7 presents another architecture for providing this diagnostic information in an unsupervised manner.

Subsequently, Part III presents the general conclusions drawn from this thesis, as well as potential future lines of research. The main achievements and contributions are also summarized in this final section. Lastly, in Part IV the appendices provide supplementary information and resources related to the topics explored throughout the thesis.

2

State of the art review

In this chapter, we explore the latest developments in Explainable AI (XAI) with an emphasis on using these models for anomaly diagnosis. Our primary goal is to use XAI models to identify and understand unusual data patterns. By examining the current best practices and methods, our aim is to connect the dots between detecting anomalies and explaining them. This exploration offers a clear understanding of current techniques, helping us move towards better and clearer anomaly diagnosis using XAI.

2.1 Explainable Artificial Intelligence

To diagnose anomalies effectively, it is essential to first delve into the realm of Explainable AI. This branch of AI focuses on making machine learning decisions transparent and understandable. In this section, we explore some of the most significant works in the field of XAI. As depicted in Figure 2.1, we have organized the literature into three main categories: pre-model interpretability, intrinsic interpretability, and post hoc interpretability. This categorization guides us in understanding the various methods available for explaining AI actions.

2.1.1 Pre-model interpretability

Pre-model interpretability, also known as data-interpretability, is typically conducted before building the model. It plays a crucial role in understanding the input data and is an important phase in any machine learning (ML) problem. The primary component of pre-model interpretability involves **performing exploratory analysis of the data** [2], which can include visualization charts, summary statistics, clustering, and more.

Data visualization is a fundamental aspect of pre-model interpretability since data with similar statistics can exhibit significant differences [3, 4]. Given that datasets often contain numerous features, dimensionality reduction

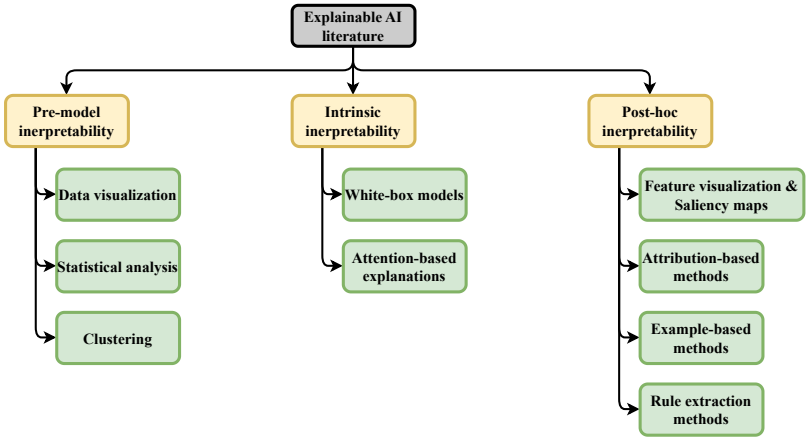


Fig. 2.1: Taxonomy of XAI literature.

techniques are commonly employed for data visualization, including Principal Component Analysis (PCA) [5], Individual Component Analysis (ICA) [6], or t-Distributed Stochastic Neighbor Embedding (t-SNE) [7]. However, t-SNE can be relatively slow, especially for large datasets. In such cases, techniques like Uniform Manifold Approximation and Projection (UMAP) [8] can be used.

Clustering algorithms are also utilized to gain insights into the data. Clustering algorithms provide an understanding of data distribution, the complexity of the dataset, and the relationships between different classes [9]. Among the commonly used clustering algorithms for extracting this information are K-means and k-Nearest Neighbors (k-NN).

While global displays of data distribution help in understanding the data on a broader scale, it is often necessary to examine individual data points as well. Since large amounts of data are typically used to train models, selecting which data to visualize can be challenging. One approach to decide which data to visualize is through **prototypes** [10] **and criticisms** [11]. Prototypes are instances that best represent the data distribution, while criticisms are data instances that deviate the most from the data distribution. Both prototypes and criticisms can be identified using greedy search, and one widely-used technique for finding these instances is the Maximum Mean Discrepancy-Citric (MMD-Citric) [11].

2.1.2 Intrinsic interpretability

Model internal interpretability comes from models that are interpretable by themselves or knowledge can be extracted from their internals, either through weight analysis, internal feature visualization, saliency maps, etc.

Interpretability referring to model internals is, by definition, intrinsic interpretability.

A. White-box models

One way to obtain intrinsic explanations is through models that are purely interpretable by themselves, which are called *white-box* models.

Among this family of models, we can find **linear models** such as Linear Discriminant Analysis (LDA) [12], Logistic Regression, and Linear Classifiers [13] whose decisions are easily interpreted, since it allows associating the input features and the output linearly: a linear change in the input feature leads to a linear change in the output.

Other methods like Decision Trees [14], Rule sets [15], or Decision sets [16] represent distinct but related rule-based methodologies. **Decision trees** operate through a hierarchical, tree-like structure where each node encapsulates a decision rule guiding the data flow. **Rule sets** employ a collection of conditional "if-then" statements to determine classification outcomes. **Decision sets** also use a list of rules, however, they incorporate an order to these rules and include an "else" clause for situations when the rule's conditions are not fulfilled. All three methodologies offer significant transparency due to their rule-centric nature

Case-based reasoning [17], Generalized Additive Models (GAMs) [18], and Interpretable Fuzzy Systems can also be considered as white boxes. **Case-based reasoning** uses memory-based problem solving, leveraging existing cases to understand new ones, thus providing transparent decision-making. **Generalized additive models** model the response variable as an interpretable sum of smooth functions of predictors, enabling them to handle complex, nonlinear relationships. **Interpretable fuzzy systems** employ fuzzy logic, sets, and rules to manage uncertainty, with their interpretability arising from their use of easily comprehensible linguistic variables and rules.

Falling Rule Lists [19] is a method designed for binary classification. This method generates outputs in the form of probabilistic if-then rules for decision-making. Unlike other rule-based approaches such as Decision Trees, the if-then rules in Falling Rule Lists are organized according to the estimated probability of success. This means that the conditions are presented in 'if' clauses and the probabilities of the desired outcome are expressed in the 'then' clauses, with these probabilities decreasing monotonically down the list.

A more recent development in this field is the **Generalized Linear Rule Models** (GLRM) [20]. GLRM combines decision rules with GAMs, integrating features of generalized linear models with additive models. Consequently, they offer both non-linear modeling capabilities through the decision rules, while also maintaining interpretability via the linear model.

Despite their inherent transparency, the explainability of white-box models is not guaranteed without the requisite of understanding how to interpret them correctly [21]. Furthermore, these models, due to their simplicity, may

not be suitable for handling non-linear complex contexts, limiting their application [22, 23, 24]. In such scenarios, more sophisticated models like Deep Learning are employed, necessitating the use of alternative techniques to evaluate interpretability.

B. Attention based explanations

Attention mechanisms elucidate the regions a model emphasizes during the prediction process, offering valuable insights into its operational dynamics. These mechanisms have been employed particularly in the context of sequential data due to the fact that attention mechanisms are commonly applied on recurrent networks or Transformers. Moreover, in the realm of image data, when applied to Convolutional Neural Networks (CNNs), attention mechanisms afford a visual mapping of the areas the model prioritizes.

Concerning sequential data interpretability, Choi et al. [25] devised an interpretable model, termed as the **REverse Time Attention (RETAIN)**, to predict the risk associated with hospital visits and event records of heart failure patients. The RETAIN model, which is based on a two-level neural attention mechanism, provides clinically interpretable outcomes without compromising on prediction accuracy. This approach employs two Recurrent Neural Networks (RNNs); one for determining patient visit-level attention and the other for variable-level attention. The patient visit-level attention weights, represented as α, act as a measure of the influence of each visit embedding. Conversely, the variable-level attention weights, denoted as β, measure the influence of each variable within the visit embedding. The versatility of the RETAIN mechanism extends beyond healthcare and has been effectively employed in time series prediction tasks [26]. Here, the attention weights α and β ascertain the contribution of each input at specific time steps. In a similar vein, Bai et al. [27] proposed the utilization of code-specific disease progression functions and attention mechanisms [28] to gauge the contribution of each medical code. This approach considers the occurrence and the nature of the medical code. Consequently, the weights corresponding to the embeddings created by the attention and the disease progression function offer a perspective on the importance accorded by the network to each code.

In the system logs anomaly detection, attention-based explanations foster a sense of trust among analysts and administrators towards the models in use. As exemplified by **DeepLog**, an LSTM-based model presented by Du et al. [29], Recurrent Neural Networks (RNNs), specifically LSTM, have proven to be effective for anomaly detection and diagnosis from system logs. Subsequently, Tuor et al. [30] proposed four distinct LSTM-based models for cyber anomaly detection, providing the foundation for Brown et al. [31] who incorporated various attention mechanisms over these models. This adaptation not only assured superior performance but also offered insights into feature importance and relational mapping between features.

As **for time series data, attention mechanisms have showcased their effectiveness in both anomaly detection and prediction tasks**.

Giurgiu et al. [32] proposed an approach for detecting anomalous events in storage environments based on Key Performance Indicators (KPIs). By leveraging an attention mechanism and a contribution function, the authors were able to represent each event window and estimate its individual contribution toward future failure predictions.

Regarding multivariate time series, Schockaert [33] presented an approach utilizing an attention mechanism and guided backpropagation for generating spatio-temporal explanations. Here, the regression model, which predicts the temperature of hot metal produced by a blast furnace, consisted of a blend of 1D CNNs and an LSTM. A backpropagation-based approach highlighted which dimensions of the input were inducing the gradient in the hidden state.

Similar spatio-temporal explanations for multivariate time-series data can also be achieved using **Grad-CAM** [34], as demonstrated by Assaf et al. [35]. Given that time-series data is often processed by RNNs, various tools have been developed to visualize attention heatmaps on RNNs, such as **Attention-Heatmaps** [36] and **ViSFA** [37], to facilitate explanations based on attention mechanisms.

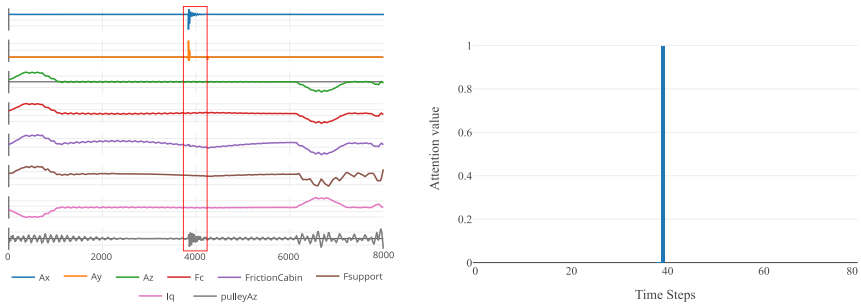
On image data, attention mechanisms applied to CNNs are gaining strength in recent years. Advancements such as **attention residual nets** [38] and **squeeze networks** [39] have demonstrated that the inclusion of attention mechanisms in image data architectures can enhance both their accuracy and interpretability.

Qin et al. [40] proposed an attention-focused convolutional layer, named **Autofocus**, for semantic segmentation. This approach parallelizes multiple CNNs with varying dilation rates, with the optimal scales being learned through an attention mechanism. The attention maps generated can facilitate the detection of patterns at different scales, since each attention map denotes the degree of focus allocated to each scale.

Recently, an attention-based methodology named **IASSA** was introduced by Vasu et al. [41]. This technique identifies crucial regions within an image and is versatile enough to be employed in any application that utilizes deep neural networks for feature extraction. The procedure involves passing an image through a black-box classifier that provides logit scores for each sample; these scores are then used to weight image regions. The derived map is combined with an attention map produced by a long-range and parameter-free spatial attention module (LRPF-SA), resulting in a saliency map that underscores the most significant regions. This attention map is employed in the subsequent iteration to sample the relevant regions, a process that is repeated until convergence.

Shitole et al. [42] present an innovative approach, termed **Structured Attention Graphs** (SAGs), which goes beyond generating a single saliency map to creating visualizations of collective attention maps for each image. This effectively illustrates the cumulative effect of varying image regions on the classifier's confidence. The incorporation of the beam search algorithm in their research further enriches interpretability by generating multiple ex-

planations for each image, thereby offering a more extensive understanding of the classification process. A user study conducted to test the efficacy of SAGs demonstrated its significant potential in enhancing user comprehension of decision-making processes within CNNs.



(a) Input data containing an anomaly in the middle of the journey.

(b) Attention activations of the input fed to the classifier.

Fig. 2.2: The attention mechanism allow us to detect where the model is focusing when classifies the journey as anomalous.

Figure 2.2a provides an illustrative example of an anomalous elevator journey. The model in use broadly categorizes the journey as either normal or anomalous without specifying the anomaly's precise location. However, with the integration of the attention mechanism, we can extract more insights. The attention weights reveal the exact location of the anomaly, as highlighted in Figure 2.2b. This demonstrates the capability of attention mechanisms to offer enhanced information about anomalies.

2.1.3 Post-hoc interpretability

A. Feature visualization and saliency maps

Due to the success of the CNNs for a variety of problems, many works have **used the filters of these layers to visualize them** and understand what the model is learning at each step [43]. The visualization techniques extend to **Global Average Pooling** (GAP) layers and **Class Activation Maps** (CAM) [44]. These techniques underscore the significant regions of the input, which proves useful for object localization tasks. Visualization of convolutional filters is particularly informative in image data analysis, offering intuitive insights into the learned features [45, 46]. Despite this, filter visualization remains less common in interpreting models in time-series data. Nonetheless, Siddiqui et al. propose a framework for decoding convolutional Deep Learning

(DL) models applied to time-series analysis in [47]. This aids in identifying predictive regions and comprehending the importance of each filter.

Deep Learning models usually feature a design where part of the model extracts characteristics from input data, and the other uses these characteristics to address the problem at hand. **Analyzing this latent space** can be informative to understand what the model has learned. In this context, Pereira et al. [48] suggested employing a variational Bi-LSTM Autoencoder with attention for anomaly detection in solar energy data. The latent space formed by the encoder is visualized using PCA and t-SNE, and different types of faults are examined within this space. t-SNE visualizations are also utilized by Xu et al. [49] for **cluster analysis of embeddings** created by different DL models in a wide-range of datasets.

In addition to model weights, gradients play a crucial role in identifying the areas upon which the model primarily focuses. These **gradient-based techniques** are calculated via the partial derivatives of the target concerning the input features. Ancona et al. [50] have performed an analysis comparing five such methods: **Saliency Maps** [51], **Gradient*Input** [52], **ϵ-LRP** [53], **DeepLIFT** [54], **Integrated Gradients** (IG) [55], and an **Occlusion** method, which is a perturbation-based approach [43].

LRP, one of the pioneering gradient-based techniques, quantifies the contribution of each individual pixel in an image to the final prediction. An evolution of such methods, **DeepLift**, traces the influence of all neurons back to each input feature, subsequently deriving an importance score by contrasting the activations within the network with a set of reference activations.

In response to the challenges of LRP and DeepLIFT, particularly their inability to ensure implementation invariance, IG was introduced. Implementation invariance refers to the necessity for identical attributions for networks that are functionally equivalent, regardless of differences in their structure. To resolve this, IG utilizes continuous gradients for attribution backpropagation, a departure from the discrete gradients used by LRP and DeepLIFT. IG achieves this by envisioning a linear path from a baseline input to the original input and cumulatively adding all gradients along this path. Specifically, integrated gradients are delineated as the path integral of gradients along the straight-line path.

Among the intrinsic gradient-based approaches, **GradCAM** [34] is widely implemented in image analysis [56, 45]. In GradCAM, the gradients of the targets are directed into the concluding convolutional layer, facilitating the generation of visual class-discriminative explanations. Figure 2.3 displays two saliency maps generated by GradCAM. These maps highlight the locations of the butterfly and flowers within the input image.

B. Attribution-based methods

Attribution-based methods aim to explain model decisions based on **the importance that each variable has had in the model output**.

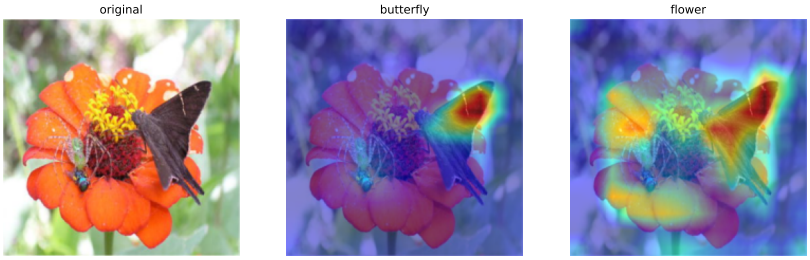


Fig. 2.3: Example of saliency maps produced by GradCAM.

LIME [57] is a widely adopted interpretability method for quantifying feature influence. Let $X = \{\mathbf{x}|\mathbf{x} \in \mathbb{R}^V\}$ be the original input space in which $\mathbf{x} \in X$ is the original representation of an instance being explained, i.e an input of the black-box $f : \mathbb{R}^V \to \mathbb{R}$, and let $X' = \{\mathbf{x}'|\mathbf{x}' \in \{0,1\}^{V'}\}$ be the interpretable feature space, in which $\mathbf{x}' \in X'$ is the interpretable representation of the instance $\mathbf{x}$, in the form of a binary vector. Using this notation, LIME performs four steps to obtain the explanations:

1. N perturbed instances are sampled around $\mathbf{x}'$ by drawing nonzero elements of $\mathbf{x}'$ uniformly at random. Let $Z = \{\mathbf{z}'_i \in X'|i = 1, ..., N\}$ be the set of these perturbed instances.
2. Recover each sample $\mathbf{x}'_i \in Z$ in the original representation $\mathbf{z}_i \in \mathbb{R}^V$.
3. For each $\mathbf{z}_i$, $f(\mathbf{z}_i)$ is obtained, which is used as a label for the explanation model.
4. Given the dataset Z of the perturbed instances with their associated labels, the explanation is given by optimizing the following equation:

$$\xi(\mathbf{x}) = \underset{g \in G}{\operatorname{argmin}}\{\mathcal{L}(f, g, \pi_x) + \Omega(g)\} \tag{2.1}$$

In some cases examining all the predictions given by a model can be difficult. Thus, a variant called SP-LIME is also proposed in [57], called SP-LIME, that uses the concept of submodular functions (functions that exhibit diminishing returns) to select a representative set of instances and corresponding explanations that together provide a global insight into the model's behavior.

In the literature, there are numerous variations of this method. For instance, Mishra et al. [58] proposed **SLIME**, a variation of LIME for generating temporal and time-frequency explanations on music content analysis tasks. Moreover, Peltola et al. [59] presented KL-LIME, an approach combining LIME with Bayesian projection predictive variable selection methods, which is used for explaining MNIST digit classifications made by a Bayesian deep convolutional neural network.

Following LIME, several other methods have emerged in the field of feature attribution. Deep Learning Important FeaTures, better known as **DeepLIFT** [54], is a model-specific gradient-based attribution method [50] that explains

the change in output from some "reference" output in terms of the change of the input from some "reference" input. Let x_i be an input neuron and x_i^0 be the "reference" input, and let t represent a target output neuron and t^0 the "reference" activation of t. Let $\Delta x_i = x_i - x_i^0$ and $\Delta t = t - t^0$ be the differences between the input and the target to their corresponding references. DeepLIFT attributes a value $C_{\Delta x_i, \Delta t}$ to each input x_i, representing the effect of the input being set to a reference value as opposed to its original value. DeepLIFT uses a *summation-to-delta* property that states:

$$\sum_{i=1}^{V} C_{\Delta x_i, \Delta t} = \Delta t \tag{2.2}$$

Since DeepLIFT is a backpropagation method, they define some multipliers that are useful during backpropagation. For an input x_i, the multiplier $m_{\Delta x_i, \Delta t}$ is the contribution of Δx_i to Δt divided by Δx_i.

$$m_{\Delta x_i \Delta t} = \frac{C_{\Delta x_i \Delta t}}{\Delta x_i} \tag{2.3}$$

The backpropagation consists of applying a chain rule to calculate the multipliers for each neuron. The authors propose a chain rule that is consistent with the summation-to-delta property. For an input neuron x_i, a hidden layer with neurons $h_1, ..., h_k$ and a target output neuron t, given the values of the multipliers $m_{\Delta x_i, \Delta h_j}$ and $m_{\Delta h_j, \Delta t}$, the chain rule for multipliers is defined as:

$$m_{\Delta x_i \Delta t} = \sum_j m_{\Delta x_i \Delta h_j} m_{\Delta h_j \Delta t}. \tag{2.4}$$

SHapley Additive exPlanations (SHAP) [60], which stands as one of the most widely adopted interpretability methods, interprets model decisions based on game theory principles. SHAP explanations employ Shapley values, viewing the black box problem through the lens of coalitional game theory. In order to understand the intuition behind how SHAP works, first Shapley values need to be defined.

Definition 25. *Let N be the set of n players, v a function that gives the value for any subset $S \subseteq N$. Then the* ***Shapley value*** *for a player i is defined as (Equation (2.5)):*

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (v(S \cup \{i\}) - v(S)). \tag{2.5}$$

Let g be the explanation model used to explain the black-box model f. Despite the global interpretation that can be deduced from local explanations, SHAP is a local method, designed to explain the prediction $f(\mathbf{x})$ given to an input $\mathbf{x} \in X$. Let $\mathbf{x}' \in X'$ be a simplified input and $h_x : X' \to X$ be a mapping function, such that $x = h_x(\mathbf{x}')$. Whenever $\mathbf{z}' \approx \mathbf{x}'$, local methods try to ensure $g(\mathbf{z}') \approx f(h_x(\mathbf{z}'))$.

Definition 26. *In **Additive feature attribution methods** the explanation model g is a linear function of binary variables:*

$$g(\mathbf{z}') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{2.6}$$

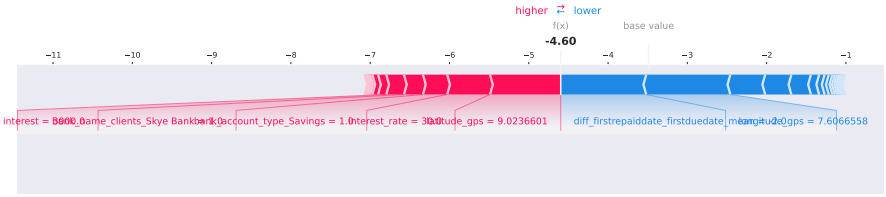
where $\mathbf{z}' \in \{0,1\}^M$ is the coalition vector, being M the number of simplified input features, and $\phi \in \mathrm{R}$.

When the coalition vector $\mathbf{z}$ equals to one indicates that the relevant feature value is "present", whereas $\mathbf{z} = 1$ indicates that the feature is "missing".

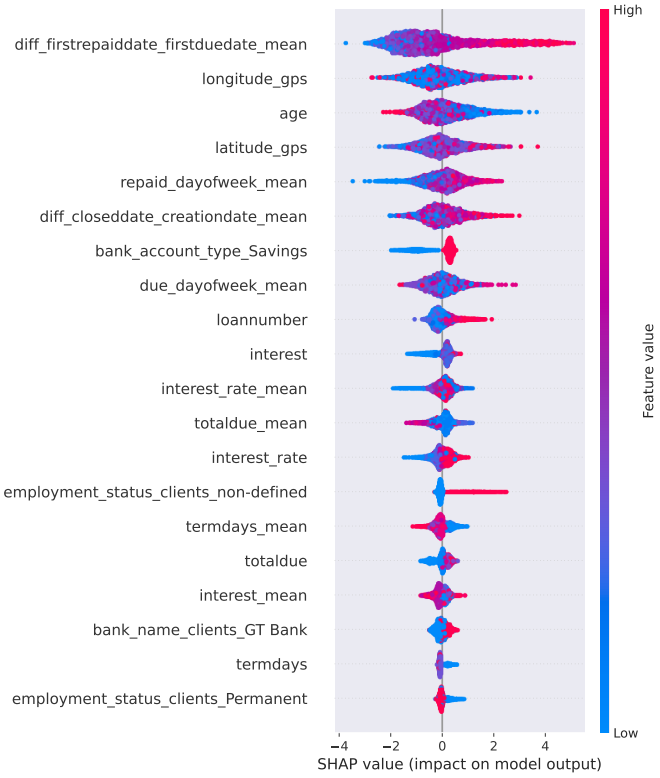
The original paper on SHAP establishes three desirable properties for feature attribution methods, with Shapley values being the only explanations capable of meeting these criteria. However, the exact computation of SHAP values can be challenging. Therefore, SHAP values are often approximated by incorporating insights from other additive feature attribution methods such as DeepLIFT, LIME, or IG. SHAP offers various approximation techniques for estimating these values. Kernel SHAP, for instance, is a model-agnostic explainer that builds upon LIME's approach. It employs specially weighted local linear regression to estimate SHAP values, enabling the interpretation of complex models across different data types, including tabular, sequential, and image data [61]. Additionally, SHAP provides a variation called TreeExplainer, designed specifically for explaining tree-based models [62].

Regarding the interpretability of DL models, SHAP provides two different explainers: **GradientExplainer** and **DeepExplainer**. On the one hand, GradientExplainer combines ideas from IG, SHAP, and SmoothGrad [63] into a single expected value equation. The entire dataset is used as a background distribution and allows local smoothing. Approximating the model with a linear function between each background data sample and the current input to be explained, and assuming that the input features are independent, SHAP values are approximated by the expected gradients. Unfortunately, GradientExplainer is slightly slow. On the other hand, DeepExplainer is a high-speed algorithm for approximating SHAP values, which is faster than GradientExplainer and works with DL models. DeepExplainer approximates SHAP values with an adaptation of DeepLIFT, which becomes a compositional approximation of SHAP values. It combines SHAP values computed for smaller components of the network into SHAP values for the whole network by recursively passing DeepLIFT's multipliers, which are defined in terms of SHAP values.

Figures 2.4a and 2.4b showcase SHAP's application in a credit risk scoring context, where diverse data inputs, from financial history to demographics, drive creditworthiness assessments. Figure 2.4a, the force plot, provides a granular analysis for an individual case. The bold value signifies the logarithm of the odds ratio, with forces in blue indicating features that decrease the probability of default, and those in red suggesting a higher risk. This plot



(a) Local Explanations.



(b) Global Explanations.

Fig. 2.4: SHAP visualizations for credit risk assessment: (a) Force plot showcasing individual predictions and (b) Summary plot revealing global model trends.

aids in understanding the specific reasons behind a model's decision on credit approval, with particular factors such as longitude and initial payment delay playing pivotal roles in the illustrated instance. Conversely, Figure 2.4b delivers a more expansive perspective. The SHAP summary plot ranks features based on their overall influence, with each dot corresponding to a distinct

individual. This global view offers insight into the overarching patterns and dominant features steering the model's decisions, notably payment delays and geographical data.

C. Example based methods

By definition, an example is a representative instance or a fragment of something, selected to illustrate the nature of the entire entity. Various methods in academic literature **employ examples to elucidate model decisions**. These examples can either be existing instances or entirely new instances.

Examples based on existing instances can either be given as **prototypes** or **criticisms**. Prototypes and criticisms are effective in comprehending the data distribution, highlighting instances that best represent the distribution and those that are the most deviant. When it comes to interpreting model decisions, they help identify examples that significantly influence a specific set of predictions. In the literature, numerous works have focused on the selection of prototypes [64, 65, 66, 67].

In the context of **prototype selection** via Deep Learning (DL) models, Li et al. [67] proposed an architecture capable of learning prototype vectors. This architecture is validated across various image-classification tasks. The proposed architecture comprises an AE, which is employed to decrease the dimensionality of input data and to learn pertinent features for prediction, and a prototype layer that learns prototype vectors. The prototype layer calculates the squared L^2 distance between the encoded input and each prototype vector. These prototype vectors are then input into a fully connected network equipped with a softmax activation function, which generates a probability distribution over all classes. The loss function, a key part of the architecture, is a sum of four terms. The first term penalizes misclassification via a cross-entropy loss between the model's output and the training data labels. The second term is derived from the Mean Squared Error (MSE) between the reconstructed samples and the original ones. The final two terms of the loss function serve as regularization terms for interpretability. The first regularization term ensures each prototype vector is as proximate as possible to at least one training example in the latent space, while the second ensures every encoded training example is as close as feasible to one of the prototype vectors. Besides image data, this architecture has been implemented by Gee et al. [68] for time-series data, where the prototypes of latent space prompted by DL models are visualized to yield explainable insights on three distinct time series classification tasks.

Continuing with methods based on examples, **counterfactual explanations** are a prevalent approach for elucidating the decision-making processes of opaque, or 'black box', models. Such explanations address what-if scenarios. Specifically, a counterfactual explanation for a prediction **outlines the minimal alteration to feature values that would result in changing the prediction to a predefined outcome**.

Counterfactual explanations were initially introduced by Wachter et al. [69], where they were characterized as adversarial perturbations [70]. Until that point, adversarial perturbations were depicted as minor adjustments applied to numerous variables with the intention of altering the classification, but not necessarily with the objective of rendering these changes interpretable to humans. This is because many of these perturbations while having a significant impact on the classifier's response, are barely noticeable to the human eye. In earlier works related to adversarial samples, the newly produced samples often do not appear to reside in the realm of real samples. Therefore, the authors highlight that incorporating suitable distance functions between the generated and real samples within the optimization problem aids in the creation of interpretable counterfactuals.

Dhurandar et al. [71] proposed a method called **Contrastive Explanations Method** (CEM). CEM is a perturbation-based model-agnostic method that provides local explanations. The method consists of solving two different optimization problems, one for finding *Pertinent Negatives (PN)* and the other for finding *Pertinent Positives (PP)*.

1. **Finding pertinent negatives:** Finding pertinent negatives consist of finding an interpretable minimal perturbation of the input that differentiates it from the closest different class Let $\mathbf{x}_0$ be an input of a black-box model f for which we want to explain the prediction $f(\mathbf{x}_0)$ and y_0 its corresponding class. Let $\mathrm{AE}(\cdot)$ be an autoencoder trained for reconstructing an input. Denoting $\mathcal{X}/\mathbf{x}_0$ to the space of missing parts with respect to $\mathbf{x}_0$, finding pertinent negatives consist of finding an interpretable minimal perturbation $\delta \in X/\mathbf{x}_0$ such that $\arg\max_i \left[f\left(\mathbf{x}_0\right)\right]_i \neq \arg\max_i \left[f\left(\mathbf{x}_0+\delta\right)\right]_i$. For finding pertinent negatives, the authors propose solving this optimization problem (Equation (2.7):

$$\min_{\delta \in \mathcal{X}/\mathbf{x}_0} c \cdot f_\kappa^{\mathrm{neg}}\left(\mathbf{x}_0, \delta\right) + \beta\|\delta\|_1 + \|\delta\|_2^2 + \gamma\left\|\mathbf{x}_0+\delta-\mathrm{AE}\left(\mathbf{x}_0+\delta\right)\right\|_2^2 \quad (2.7)$$

where $c, \beta, \gamma \geq 0$ are regularization parameters. The second and the third terms, $\beta\|\delta\|_1$ and $\|\delta\|_2^2$ respectively, are jointly called the elastic net regularizer and it is used for efficient feature selection in high-dimensional learning spaces. The third term $\gamma\left\|\mathbf{x}_0+\delta-\mathrm{AE}\left(\mathbf{x}_0+\delta\right)\right\|_2^2$ ensures that the modified input $x_0+\delta$ is close to the data manifold. The first term $f_\kappa^{\mathrm{neg}}\left(\mathbf{x}_0, \delta\right)$ is defined in this way:

$$f_\kappa^{\mathrm{neg}}\left(\mathbf{x}_0, \delta\right) = \max\left\{\left[f\left(\mathbf{x}_0+\delta\right)\right]_{y_0} - \max_{i \neq y_0}\left[f\left(\mathbf{x}_0+\delta\right)\right]_i, -\kappa\right\} \quad (2.8)$$

where $\left[f\left(\mathbf{x}_0+\delta\right)\right]_i$ is the score given by the model f to the i-th class prediction. Introducing $f_\kappa^{\mathrm{neg}}\left(\mathbf{x}_0, \delta\right)$ to the Equation 2.7, ensures $x_0+\delta$ to be predicted as a different class than y_0. The parameter $\kappa \geq 0$ is a confidence parameter to control the separation between $\left[f\left(\mathbf{x}_0+\delta\right)\right]_{y_0}$ and $\max_{i \neq y_0}\left[f\left(\mathbf{x}_0+\delta\right)\right]_i$.

2. **Finding pertinent positives:** Finding pertinent positives consist of finding an interpretable perturbation, such that removing it from the original input the class does not change For finding pertinent positives, let $\mathcal{X} \cap x_0$ be the space of existing components of x_0 and let $\delta \in \mathcal{X} \cap x_0$ be an interpretable perturbation such that removing it from x_0 the prediction is still the same, i.e $\arg\max_i [f(x_0)]_i = \arg\max_i [f(\delta)]_i$. To this end, similar to finding pertinent negatives, we need to solve the following optimization problem:

$$\min_{\delta \in \mathcal{X} \cap x_0} c \cdot f_\kappa^{\text{neg}}(\mathbf{x}_0, \delta) + \beta \|\delta\|_1 + \|\delta\|_2^2 + \gamma \|\delta - \text{AE}(\delta)\|_2^2 \tag{2.9}$$

where the first term $f_\kappa^{\text{neg}}(\mathbf{x}_0, \delta)$ is defined in this way:

$$f_\kappa^{\text{neg}}(\mathbf{x}_0, \delta) = \max \left\{ \max_{i \neq y_0} [f(\delta)]_i - [f(\delta)]_{y_0}, -\kappa \right\} \tag{2.10}$$

To solve the optimization problems (2.7) and (2.9), a projected fast iterative shrinkage-thresholding algorithm, called FISTA [72], is used.

Combining prototypes with counterfactuals, Van et al. [73] proposed **CFPROTO** and showed that the use of prototypes can speed up the search for counterfactual instances. In this work, the authors present a variation of the search for pertinent negatives in CEM [71], which are counterfactual explanations. Here the loss function used is the same as the one defined in Equation (2.7), but a new term, denoted as L_{proto}, associated with prototypes is added. The prototypes in this work are defined as the average encoding over the K nearest instances belonging to the same class label, denoting as proto_i the prototype belonging to class i. Following the same notation as in CEM, the Equation (2.7) now becomes in

$$\min_{\delta \in \mathcal{X}/x_0} c \cdot f_\kappa^{\text{neg}}(x_0, \delta) + \beta \|\delta\|_1 + \|\delta\|_2^2 + L_{AE} + L_{proto} \tag{2.11}$$

where

$$L_{AE} = \gamma \|x_0 + \delta - \text{AE}(x_0 + \delta)\|_2^2 \tag{2.12}$$

and L_{proto} is defined as follows:

$$L_{proto} = \theta \cdot ||ENC(x_o + \delta) - \text{proto}_j||_2^2 \tag{2.13}$$

where ENC denotes an encoder that projects each data onto a lower dimensional latent space. The new term L_{proto} guides the perturbations towards the nearest proto_j, where $j \neq y_o$ denotes the nearest prototype of class j to the encoding of x_0, which speeds up the counterfactual searching process.

Counterfactual explanations have also been given using SHAP. Rathi et al. [74] give the explanations in response to "Why [predicted-class] (P) not [desired-class] (Q)?" For this, they calculate the Shapley values for each possible class. The explanations are given as text, so they divide the

question to be answered into two parts: “Why P?” and “Why not Q?”. The answer for these two segments is constructed using the Shapley values for class P and class Q and counterfactuals are obtained by mutating the features that work against the classification of the desired category.

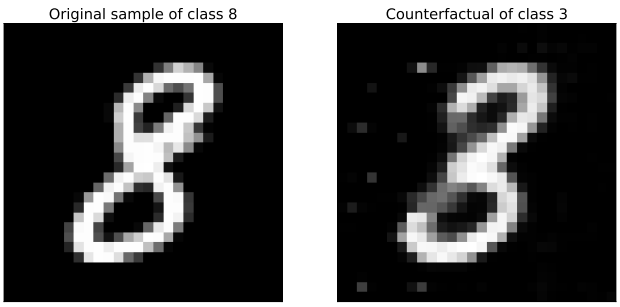


Fig. 2.5: Example of a counterfactual explanation applied to MNIST dataset.

Figure 2.5 presents an example of a counterfactual explanation applied to a sample from the MNIST dataset, which represents the digit ”8”. This explanation elucidates the minimal alterations on that sample needed for the model to modify its prediction from the digit ”8” to the digit ”3”.

D. Rule extraction methods

Explaining model decisions in the form of decision rules is another common approach. The **RuleFit** algorithm [75] is a regression and classification method that incorporates decision rules within a linear model. RuleFit has two main components: the first is responsible for generating rules, and the second is for training a linear model using these rules.

Initially, the Gradient Boosting technique [76] is utilized to fit an ensemble of Decision Trees, from which multiple rules are extracted by converting each path in a tree into a decision rule. To minimize the number of rules produced from the tree ensembles, these rules are further condensed in the second step using linear models like Lasso or L1 regularized regression. In this step, each rule is treated as a feature and assigned a weight estimate, with many estimates defaulting to zero.

Skope Rules offer another approach for extracting rules using tree ensembles. However, the rule selection in Skope Rules is based on recall and precision thresholds. This means that Lasso is not used for rule selection. Instead, only the out-of-bag F1-score and the logical terms that compose the rules are considered.

Anchors [77], is one of the most popular rule-based algorithm. It is used for interpreting black-box models and it is a model-agnostic method. Anchor was proposed for addressing an issue regarding local explanation methods like

LIME, which represent the local behavior of the model in a linear fashion. In [77], an interesting example of how LIME and Anchors explain the outcome given by an LSTM in a sentiment analysis task is given. The example is analyzing if each word in these two sentences is positive or negative:

+ This movie is not bad.

- This movie is not very good.

Using LIME, for example, the influence that the "not" has had in the first case is not applicable to the influence that it has had in the second case, so with the explanations of LIME it is unclear when the word "not" has a negative influence and when it has a positive influence. In Anchors, the explanation for the word "not" is not given alone, instead, the explanations are given in this form:

{"not","bad"} → Positive ; {"not","good"} → Negative

To obtain this type of explanation, first, the candidate rules are chosen so that they can explain a data point. Then, permutations are made on the data point to be explained so that these permutations do not alter the output of the model to be explained. Finally, the chosen rules are evaluated. To optimize the search and evaluation of these rules, a reinforcement learning approach, *Multi-Armed Bandit (MAB)* [78], is used, which assigns a payoff to the candidate rules according to predefined convergence criteria, filtering the rules according to a precision threshold and selecting the rules with the highest coverage.

Guidotti et al. [79] introduced an algorithm known as **Local Rule-Based Explanations (LORE)**. LORE, a model-agnostic rule-based method, surpasses both LIME and Anchors in performance and explanatory clarity. LORE constructs an interpretable predictor to explain a black box decision for a specific instance. This is achieved by generating a set of neighboring instances around the instance to be explained using a genetic algorithm and subsequently deriving a decision tree from this set. From this tree, decision rules are extracted to elucidate the black box decision, and a set of counterfactual rules are presented.

Unlike LIME, where the number of features composing an explanation is a user-specified input parameter, LORE automatically provides only the features necessary for explaining the black box. Furthermore, the use of decision trees for rule generation enables the splitting of continuous features. This overcomes the issue with Anchors and continuous features, which necessitates prior discretization.

2.2 Anomaly detection

In the pursuit of diagnosing anomalies, understanding the underlying methods of anomaly detection is crucial. Anomaly detection involves identifying

patterns in data that do not conform to expected behavior. Within this section, we will delve into some of the significant works in the domain of anomaly detection. As illustrated in Figure 2.6, the literature can be broadly divided into four primary categories: classification methods, forecasting methods, reconstruction methods, and clustering methods. This structure aids us in comprehending the diverse techniques employed to detect anomalies.

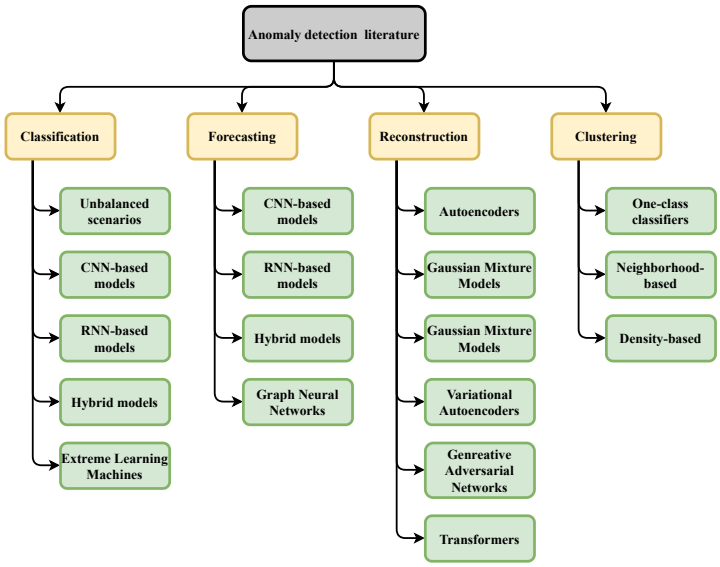


Fig. 2.6: Taxonomy of anomaly detection literature.

2.2.1 Classification methods

Supervised techniques, utilized when labeled data is accessible, involve learning a mapping between input features and corresponding labels. While supervised AD resembles a standard classification problem, it **often faces unbalanced data in AD scenarios due to the difficulty and expense of acquiring anomaly data**. This imbalance necessitates addressing biased performance towards the majority class in supervised unbalanced situations. A common solution involves **adjusting data distribution** [80, 81]. This can be achieved through **oversampling techniques**, which increase minority class data via random duplication or methods like **SMOTE** [82], **ADASYN** [83], or **RACOG** [84], among others. Alternatively, reducing majority class data balances classes in unbalanced scenarios through **random deletion of samples** [85] or **undersampling techniques** like **NearMiss** [86] or **Condensed Nearest Neighbor Rule** [87]. However, applying these techniques

to time-series data, particularly **synthetic data generation**, remains complex [88, 89, 90].

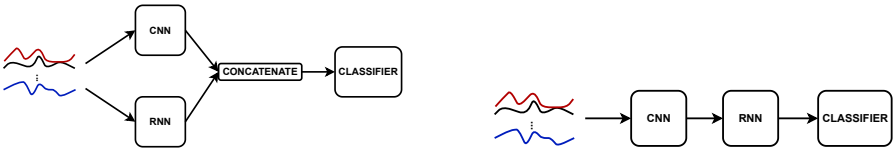
Handling unbalanced scenarios with supervised algorithms poses difficulties; however, numerous approaches have been developed to tackle these challenges. Among them are **cost-sensitive decision tree ensembles** [91], which construct classifiers based on a cost matrix to account for class imbalance. Park et al. [92] also use **decision tree ensembles**, applying an alpha-divergence splitting criterion and a lift-aware stopping criterion to minimize tree correlation. **SVM ensembles** have also been applied in anomaly detection [93]. Additionally, other methods **combine cost-sensitive learning and sampling techniques** [94, 95, 96] within a gradient boosting framework, which has been employed in AD scenarios [97].

Another approach frequently used when working with unlabeled or partially labeled data are the **self-supervised approaches**. Mahmoud et al. [98] proposed a two-stage anomaly detection approach combining self-supervised with unsupervised approaches to detect cyber physical attacks in water distribution systems. Blazquez-Garcia et al. [99] use a self-supervised methodology for detecting water leaks within a water distribution company. In this methodology, they apply a time series classification algorithm called **Random Interval Spectral Ensemble** (RISE) [100], resulting in an optimal equilibrium between accurately identifying water leaks and having a high true positive rate. In [101] **TimeAutoAD** is proposed, an algorithm that uses a self-supervised contrastive loss approach to perform autonomous anomaly detection in multivariate time series. Ircio et al. [102] propose a methodology for hard drive degradation detection that operates in two rounds: in the first round, they train a classifier using labeled data under two assumptions, that the initial time windows represent normal behavior, and for failed disks, the final time window shows anomalous behavior. Subsequently, in the second round, they apply the classifier learned in the first step to identify and label the remaining windows with signs of malfunction. Finally, the final classifier is trained using all the labeled windows. It should be noted that both classifiers are optimized to minimize the minimum of the recalls between the two classes for dealing with the imbalanced situation.

When working with multivariate time series data, feature extraction or selection is often required before training classification models [103]. These transformations tend to be complex and time-consuming, necessitating expert knowledge to comprehensively understand the contribution of each variable. In contrast, deep learning (DL) models can be directly applied to raw data, as they perform these transformations automatically and efficiently. Given a raw time-series $\mathsf{X} \in \mathcal{X}$, a transformation function $T : \mathcal{X} \rightarrow \mathcal{F}$ is applied to obtain a set of features $\mathsf{F} \in \mathcal{F}$. Then, a classification function $C : \mathcal{F} \rightarrow \{0, 1\}$ is used to map the features F to their corresponding predicted labels, where 0 denotes a normal instance and 1 represents an anomaly:

$$C(T(\mathsf{X})) = \begin{cases} 0, & \text{if } \mathsf{X} \text{ is a normal instance,} \\ 1, & \text{if } \mathsf{X} \text{ is an anomaly.} \end{cases} \tag{2.14}$$

One approach to **extract features is by applying 1D convolutions**. As adjacent temporal readings are likely correlated, 1D convolutions in multivariate time series can capture local dependencies, applied either across all dimensions simultaneously [104] or individually [105]. Canizo et al. [106] found that applying 1D convolutions to each sensor separately yields better results for heterogeneous multi-sensor data, as using identical filters for different sensors may lead to suboptimal performance. Analyzing each sensor independently also facilitates network adaptation to new sensor configurations using transfer learning [107]. However, this approach may not capture the correlation between sensors. Cha et al. [108] employed **convolutional neural networks** (CNNs) as feature extractors, **followed by multilayer perceptrons** (MLPs) for crack damage detection. Similarly, 1D CNNs combined with MLPs have been utilized for time series anomaly detection [109] and cyber attack detection [110]. Ren et al. [111] proposed a method for training a CNN model in a self-supervised manner. This approach involves the use of the Spectral Residual algorithm to generate saliency maps, which are then used to identify anomalies within a sequence based on a predefined threshold. Following this, the identified data is relabeled, and a CNN is employed for the task of classifying these points.



(a) CNN-RNN in parallel. (b) CNN-RNN serially.

Fig. 2.7: CNN-RNN architectures.

While **CNNs are able to capture spatial relationships, RNNs perform better in capturing temporal relationships**, so, the combination of both is reaching great results in processing multivariate time-series data. In a parallel architecture (Figure 2.7a), spatial and temporal features are concurrently extracted via CNNs and RNNs respectively, then merged. Conversely, in a serial architecture (Figure 2.7b), CNNs first extract spatial features, which then undergo temporal extraction via RNNs. Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), or their bi-directional variants (Bi-LSTM, Bi-GRU), are commonly implemented as the RNNs in these structures. The serial CNN-LSTM architecture is prevalent in applications such as multimodal wearable activity recognition [112], automated arrhythmia diagnosis [113], gait anomaly detection in Parkinson's disease [114], and

web traffic anomaly detection [115]. Canizo et al. [106] recommend eschewing traditional RNNs in favor of LSTMs, GRUs, or their bi-directional variants, however, the optimal RNN type remains ambiguous. For instance, Xie et al. [116] report superior performance using GRUs over LSTMs in anomaly detection for industrial control systems. Liu et al. [117] propose using a hybrid network composed of a bidirectional LSTM with attention and 1D CNN with attention. These two networks are used serially as shown in Figure 2.7b, but in this case, the recurrent part is applied first. Alternatively, parallel CNN-RNN architectures have demonstrated success in predicting price development [118] and time-series classification tasks. Du et al. [119] incorporated an attention mechanism into the LSTM, a promising approach that appears to yield promising results in Natural Language Processing tasks [120] and supervised anomaly detection tasks.

Classification algorithms for AD also include **Extreme Learning Machines (ELM)**, an algorithm used to train **Single Layer Feedforward Networks (SLFN)**, in which the input weights and biased are randomly assigned and the output weights are approximated by solving a linear system mapping the inputs with the targets. Imamverdiyev et al.[121] use ELMs in network traffic AD to detect network attacks. In their work, three different activation functions have been used for ELMs: radial basis activation function (RBF), triangular basis activation function (Tribes), and Gaussian activation function with different parameter values sigma and mu. The results obtained in the NSL-KDD dataset [122] demonstrate that the ELM method provides an acceptable quality of attack classification. Others, like Wang et al. [123] propose to use L1-norm minimization ELMs to avoid overfitting, since due to the randomness of input weights many neurons may be closely correlated. With L1-norm minimization ELM, the output weights vector of the hidden layer are more sparse, which helps reducing the complexity of the model. For intrusion detection in Big Data scenarios, Xiang et al. [124] propose **MR-ELM**, a distributed implementation of ELM. Recently, Iturria et al. [125] proposed a framework that adapts predictions algorithms into anomaly detection algorithms, specifically, it proposes EORELM-AD, an ensemble of the forecasting algorithm **Online Recurrent Extreme Learning Machine** (OR-ELM) [126] adapted to AD.

2.2.2 Forecasting methods

As previously stated, **forecasting methods** are commonly employed for anomaly detection because they **do not require labels**. Deep learning techniques are typically applied to process time series data using one-dimensional convolutional neural networks (1D CNNs), recurrent neural networks (RNNs), or a combination of both. These networks can be used for unsupervised anomaly detection in time-series data. Munir et al. [127] introduced **DeepAnT**, which is an architecture that utilizes several 1D CNN layers followed by a fully connected network (FCN) that predicts the next value in a time series.

The Euclidean distance between the estimated point and the actual value is then computed, and the threshold for determining whether it is an anomaly or not is determined based on the type of time series, using either the k-σ deviation rule or the density distribution [128]. For increasing the receptive field of the convolutions within a sequence, He et al. [129] propose using dilated convolutions in a forecasting model called **Temporal Convolutional Network (TCN)**. The anomaly score proposed in this work is based on the reconstruction error and the anomalies are detected using the threshold that maximizes the harmonic mean between precision and recall.

The most common forecasting models used for anomaly detection are the ones **based on RNNs**. Particularly, LSTM networks have shown to be effective in many domains [130, 131]. Chauhan et al. [132] were one of the first ones to use LSTM networks for anomaly detection in ECG signals. The threshold employed in their work was selected based on F-score maximization. Similarly, Qin et al. [133] proposed an LSTM-based architecture for detecting abnormal traffic events. In this case, they employ a 5-σ deviation rule for selecting the anomaly threshold based on the reconstruction error. LSTM networks have also been shown to be effective in detecting spacecraft anomalies [134]. Although their method is applied to multivariate telemetry data, each channel is treated independently, training multiple LSTM networks. For detecting channel-wise anomalies, they propose using a dynamical threshold over the smoothed reconstruction error obtained using exponentially-weighted moving average (EMWA) [135]. Instead of calculating a threshold independently for each sensor, Goh et al. [136] propose applying a unique threshold over the cumulative sum of the errors. Ding et al. [137] propose a hybrid model combining LSTM and Gaussian Mixture Models (GMM). The LSTM is first used to detect anomalies in univariate data and the GMM is used to give a multidimensional joint detection of possible anomalies.

The combination of CNNs and RNNs has been also employed in forecasting models for anomaly detection. Sun et al. [**?**] use CNNs and a bidirectional LSTM with attention applied serially to process the input time-series window by window and predict the next window within a sequence. Then, the root-mean-squared error is applied as the anomaly score and, based on a predefined threshold, they detect anomalies in vehicle network data. Zhong et al. [138] proposed the use of **ConvLSTM** networks, which are similar to LSTMs but include internal convolutional operations.

To conclude the discussion on forecasting models, it is important to highlight the use of **Graph Neural Networks** (GNNs). These are **particularly beneficial when dealing with multivariate time-series data as they effectively model spatial dependencies among sensors**. Deng et al. [139] introduced the **Graph Deviation Network** (GDN), a model that leverages deviations from learned inter-sensor relationships for anomaly detection. The GDN is structured around three key components: sensor embedding, graph structure learning, and a graph attention network.

- **Sensor embedding:** This extracts embeddings from multivariate time-series data on a sensor-by-sensor basis.
- **Graph structure learning:** This forms a directed graph that uncovers the relationships between sensors.
- **Graph attention network:** This forecasts future sensor values based on a graph attention function applied over its neighbors.

Then, anomaly detection is done based on the reconstruction error obtained for each time-step and each sensor.

Chen et al. proposed **GTA** [140], a model that merges a Transformer with a graph-based learning architecture. This fusion aims to detect anomalies within multivariate time series by incorporating a graph convolution structure that simulates the influence propagation process within a network. To integrate global information into the analysis, GTA replaces the standard multi-head attention with a multi-branch attention mechanism. This mechanism is a combination of global-learned attention, conventional multi-head attention, and neighborhood convolution. Through this modification, GTA can perform a more detailed analysis of data.

2.2.3 Reconstruction methods

As stated before, the **reconstruction-based methods consist of reconstructing certain values (or all in the case of autoencoders) and comparing the input values with the reconstructed ones**.

Starting with the works that use **AEs** for this purpose, Sakurada et al. [141] used a simple fully-connected AE to perform anomaly detection in multivariate data, improving the results obtained with other dimensionality reduction techniques like PCA. However, this approach does not take into account the temporality of the data. To include temporality in the model, Malhotra et al. [142] proposed using an AE based on LSTMs. Then, they define an anomaly score based on the reconstruction error and determined the threshold that maximizes the $F_\beta = \left(1+\beta^2\right) \times P \times R / \left(\beta^2 P + R\right)$ metric, where P represents precision, R represents recall, and β is a parameter that depends on the fraction of anomalous data.

The **Deep Autoencoding Gaussian Mixture Model** (DAGMM) [143] is a more complex model that combines AEs with Gaussian Mixture Models to detect anomalies in multivariate time-series data. The model consists of two sub-networks, a compression network, and an estimation network. The compression network is a deep AE composed of an encoder $h(\cdot)$ and a decoder $g(\cdot)$, that, given an input $\mathbf{x}$, computes the reconstruction $\mathbf{x}'$ and a lower dimensional representation $\mathbf{z}$ as follows:

$$\mathbf{z}_c = h\left(\mathbf{x}; \theta_e\right) \quad \text{and} \quad \mathbf{x}' = g\left(\mathbf{z}_c; \theta_d\right) \tag{2.15}$$

$$\mathbf{z}_r = f\left(\mathbf{x}, \mathbf{x}'\right) \tag{2.16}$$

$$\mathbf{z} = \left[\mathbf{z}_c, \mathbf{z}_r\right] \tag{2.17}$$

where θ_e and θ_d are the parameters of the encoder and the decoder, respectively, $\mathbf{z}_c$ is the reduced low-dimensional representation, and $\mathbf{z}_r$ includes the features derived from the reconstruction error. Then, the estimation network consists of a GMM trained to minimize the following objective function:

$$J\left(\theta_e, \theta_d, \theta_m\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(\mathbf{x}_i, \mathbf{x}'_i\right) + \frac{\lambda_1}{N}\sum_{i=1}^{N} E\left(\mathbf{z}_i\right) + \lambda_2 P(\hat{\boldsymbol{\Sigma}}). \tag{2.18}$$

where N denotes the number of samples in the dataset. The first term of the objective is the L_2 norm between the input $\mathbf{x}$ and the reconstruction $\mathbf{x}'$, i.e. $L\left(\mathbf{x}_i, \mathbf{x}'_i\right) = \|\mathbf{x}_i - \mathbf{x}'_i\|_2^2$. The second term $E(\mathbf{z}_i)$ models the probabilities that we could observe the input samples. And the last term, defined as $P(\hat{\boldsymbol{\Sigma}}) = \sum_{k=1}^{K}\sum_{j=1}^{d}\frac{1}{\hat{\boldsymbol{\Sigma}}_{kjj}}$ (d denotes the number of dimensions in the low-dimensional representation), penalizes the small values on the diagonal of the covariance matrix, to avoid trivial solutions. The main drawback of DAGMM is that the temporality is not taken into account.

Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) [144] is another popular reconstruction-based algorithm that uses signature matrices to encode inter-sensor relationships in multivariate time-series data with a convolutional encoder and uses an attention-based ConvLSTM to incorporate temporal patterns. MSCRED is used for three tasks: anomaly detection, anomaly diagnosis, and anomaly severity. The signature matrix used in this work is a matrix that measures the inter-sensor correlations in a particular time window of length w. For two sensors i and j and their corresponding time window $\mathbf{x}_i^w = \{x_i^{t-w}, ..., x_i^t\}$ and $\mathbf{x}_j^w = \{x_j^{t-w}, ..., x_j^t\}$, respectively, their correlation $m_{ij}^t \in M^t$ is computed as:

$$m_{ij}^t = \frac{\sum_{\delta=0}^{w} x_i^{t-\delta} x_j^{t-\delta}}{\kappa} \tag{2.19}$$

being κ a reescaling factor ($\kappa = w$). In their work, they use three different signature matrices with lengths $w = 10, 30, 60$ for each time-step t. Using these matrices makes the method to be more robust to noise. Moreover, using these three matrices with short ($w = 10$), medium ($w = 30$), and large ($w = 60$) duration, the method is able to distinguish the anomaly severity.

UnSupervised Anomaly Detection for multivariate time-series (USAD) [145] is another method that uses AEs for anomaly detection. In this case, the model is composed of an encoder (E) and two decoders (D_1 and D_2), forming the autoencoders AE_1 and AE_2, which share the same encoder network. The system is trained in two phases. In phase 1, each AE minimizes its reconstruction error between an input window W and its reconstruction $\widehat{W}$, and in phase 2, AE_1 tries to match its output to the real data, while AE_2 tries to discern the real data from AE_1's reconstructions. AE_2 does not strictly act as a discriminator in the GAN sense, as its objective changes depending

on whether its input is original or reconstructed data. For anomaly detection, the following anomaly score is proposed:

$$A(\widehat{W}) = \alpha \left\| \widehat{W} - AE_1(\widehat{W}) \right\|_2 + \beta \left\| \widehat{W} - AE_2\left(AE_1(\widehat{W})\right) \right\|_2 \tag{2.20}$$

where $\alpha + \beta = 1$ are used to parameterize the trade-off between false and true positives.

Regarding **Variational Autoencoders** (VAEs), Park et al. [146] proposed using an LSTM-VAE to detect anomalies in the sensor data collected from a robot-assisted feeding system. The proposed framework uses LSTM networks for both the encoder and the decoder networks. Using the assumption that in the anomalous samples, the reconstruction probability will be lower, the anomaly score (f_s) used is the negative log-likelihood of an observation with respect to the reconstructed distribution, i.e. $f_s(\mathbf{x}_t, \phi, \theta) = -\log p(\mathbf{x}_t; \mu_{\mathbf{x}_t}, \Sigma_{\mathbf{x}_t})$.

Xu et al.[147] propose **Donut**, a VAE that tackles the problem of time-series data with missing values. Instead of using synthetic data for inputting the missing values, they treat these values as zeros, and they use M-ELBO as the training objective, which is a modified version of ELBO that excludes the contribution of anomalies and missing points. However, Donut is not a sequential model and cannot deal with temporal information. Thus, as an alternative solution that can handle temporal information, Chen et al. presented Buzz [148]. Buzz is composed of three networks that are trained adversarially. First, a variational network is used to find the pattern $q_\theta(\mathbf{z}|\mathbf{x})$ given a window $\mathbf{x}$. Then, a generative network (G) is designed to obtain the reconstruction, by applying fully connected layers and transposed convolutions. After that, a discriminative network distinguishes between real windows $\mathbf{x}$ and the reconstructed ones $G(\mathbf{z})$.

Su et al. [149] proposed **OmniAnomaly**, a stochastic recurrent neural network for multivariate anomaly detection. OmniAnomaly consists of two networks: an inference network $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ (*qnet*) and a generative network $p_\theta(\mathbf{x}_t|\mathbf{z}_t)$ (*pnet*). In this approach, the output of the inference network, denoted as $\mathbf{z}_t^0$, follows a diagonal Gaussian distribution sampled from $\mathcal{N}(\mu_{\mathbf{z}_t}, \sigma_{\mathbf{z}_t}^2 \mathbf{I})$, which is a common practice in VAEs [150]. However, to handle non-Gaussian posterior densities, the authors employ the *planar normalizing flow* (NF) [151], allowing the transformation of $\mathbf{z}_t^0$ through a series of invertible mappings. The stochastic variable $\mathbf{z}_t$ is obtained by applying K transformations f^k to $\mathbf{z}_t^0$. Additionally, to establish temporal dependencies among the $\mathbf{z}$-space variables in the inference network, the authors utilize the **Linear Gaussian State Space Model** (LG-SSM) [152]. Specifically, $\mathbf{z}_t$ is computed as $\mathbf{z}_t = \mathbf{O}_\theta(\mathbf{T}_\theta \mathbf{z}_{t-1} + \mathbf{v}_t) + \epsilon_t$, where $\mathbf{T}_\theta$ and $\mathbf{O}_\theta$ are transition and observation matrices, and $\mathbf{v}_t$ and ϵ_t represent transition and observation noises. For AD, the reconstruction probability is utilized as the anomaly score. Considering that OmniAnomaly takes a sequence $\mathbf{x}_{t-T:t}$ of T consecutive multivariate observations as input, the posterior probability at time t can

be calculated as $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-T:t} \sim \mathcal{N}(\mu_{\mathbf{x}_t}, \sigma^2_{\mathbf{x}_t}\mathbf{I}))$. Therefore, following the suggestion in [153], the reconstruction probability can be evaluated through the conditional probability $\log(p_\theta(\mathbf{x}_t|\mathbf{z}_{t-T:t}))$. Consequently, the anomaly score at time t is denoted as $S_t = \log(p_\theta(\mathbf{x}_t|\mathbf{z}_{t-T:t}))$, which is then utilized for AD by employing a threshold defined using the *Extreme Value Theory* [154].

Other **generative networks** like **GANs** have also been employed for anomaly detection. Zhou et al. [155] introduced **BeatGAN**, an adversarial autoencoder composed of an encoder G_E and a decoder G_D. The generator G_D encodes the input x into a hidden vextor z and then G_D uses z to generate the reconstruction x'. Then a discriminator D is used to play a two-player game with the generator. While the discriminator tries to distinguish real samples from synthetic samples, the generator tries to fool the discriminator by generating realistic samples. The discriminator aims to optimize the following loss function

$$L_D = \mathbb{E}x \sim P_r[\log D(x)] + \mathbb{E}z \sim P_z[\log(1 - D(G(z)))], \tag{2.21}$$

This function seeks to differentiate between the original data x and the generated data x', with different class labels 0 and 1. On the other hand, the generator G optimizes its own loss function L_G:

$$L_G = \mathbb{E}_{z \sim P_z}[\log(1 - D(G(z)))] \tag{2.22}$$

The goal of the generator is to produce data that cannot be distinguished by $D(\cdot)$, effectively creating output that closely aligns with class label 1. Since the model is trained using normal data, when an anomaly occurs the reconstruction error is higher and this serves as an anomaly score. Similarly, Chen et al. [156] proposed **DAEMON**, an adversarial autoencoder that uses two discriminators to adversarially train an autoencoder to learn the normal pattern of multivariate time series and then use the reconstruction error to detect the anomalies. The architecture of DAEMON has three parts: an autoencoder G_A that consists of an encoder G_E and a decoder G_D, a discriminator D_E, and another discriminator D_D. The key difference here is that one discriminator D_E is used to guide the posterior $q(z)$ to match a prior $p(z)$ while the encoder G_E tries to fool the discriminator. And the other discriminator D_D is used to distinguish realistic and synthetic data, while the generator G_D tries to fool D_D.

The use of **Transformer models** for anomaly detection is an emerging area [157]. Recently, Xu et al. propose Anomaly Transformer [158], a model that employs a modified version of the self-attention mechanism known as Anomaly-Attention, which includes two branching structures that model prior and serial associations at each time point respectively. The Anomaly-Attention in the l-th layer is formulated as:

$$\begin{aligned}
\text{Initialization:} \quad & \mathcal{Q}, \mathcal{K}, \mathcal{V}, \sigma = \mathcal{X}^{l-1} W_{\mathcal{Q}}^{l}, \mathcal{X}^{l-1} W_{\mathcal{K}}^{l}, \mathcal{X}^{l-1} W_{\mathcal{V}}^{l}, \mathcal{X}^{l-1} W_{\sigma}^{l} \\
\text{Prior-Association:} \quad & \mathcal{P}^{l} = \text{Rescale}\left(\left[\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{|j-i|^2}{2\sigma_i^2}\right)\right]_{i,j \in \{1,\cdots,N\}}\right) \\
\text{Series-Association:} \quad & \mathcal{S}^{l} = \text{Softmax}\left(\frac{\mathcal{Q}\mathcal{K}^{\mathrm{T}}}{\sqrt{d_{\text{model}}}}\right) \\
\text{Reconstruction:} \quad & \widehat{\mathcal{Z}}^{l} = \mathcal{S}^{l}\mathcal{V},
\end{aligned}$$

where $\mathcal{Q}, \mathcal{K}, \mathcal{V} \in \mathbb{R}^{N \times d_{\text{model}}}, \sigma \in \mathbb{R}^{N \times 1}$ are the query, key, and values, respectively, and $W_{\mathcal{Q}}^{l}, W_{\mathcal{K}}^{l}, W_{\mathcal{V}}^{l} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}, W_{\sigma}^{l} \in \mathbb{R}^{d_{\text{model}} \times 1}$ represent the parameter matrices for $\mathcal{Q}, \mathcal{K}, \mathcal{V}, \sigma$ in the l-th layer respectively. After this, the model calculates an Association Discrepancy

$$\text{AssDis}(\mathcal{P}, \mathcal{S}; \mathcal{X}) = \left[\frac{1}{L}\sum_{l=1}^{L}\left(\text{KL}\left(\mathcal{P}_{i,:}^{l} \| \mathcal{S}_{i,:}^{l}\right) + \text{KL}\left(\mathcal{S}_{i,:}^{l} \| \mathcal{P}_{i,:}^{l}\right)\right)\right]_{i=1,\cdots,N}, \quad (2.23)$$

where L denotes the number of layers, KL is the Kullback-Leiber divergence. The association discrepancy quantifies the distance between prior and serial association at each point and acts as an anomaly score. The assumption is that anomalous associations are more likely to be noticed at adjacent time-points, and thus, the anomalies will have smaller discrepancies.

TransAnomaly [159] is a model that combines VAEs with a Transformer structure. The principal objective of this fusion is to boost parallelization within the model, and subsequently reduce the computational cost of training. Notably, this approach leads to an almost 80% reduction in training costs. Thus, TransAnomaly serves as a powerful tool for optimizing computational resources and improving the efficiency of training complex machine learning models.

MT-RVAE, introduced by Wang et al. [160], uses a multiscale Transformer to process time-series data. This model's main goal is to extract and integrate information from time-series data at various scales. Unlike many conventional Transformer models, MT-RVAE is designed to handle data at multiple levels of scale. As such, it provides a more comprehensive approach to time-series analysis. Furthermore, in scenarios characterized by low dimensionality or sparse interdependencies amongst sequences, MT-RVAE modifies the positional encoding module and introduces a feature-learning module.

Tuli et al. [161] recently introduced **TranAD**, a sophisticated transformer network architecture that incorporates two encoder-decoder pairs. For a given time instance t, TranAD operates with an input window W_t of length K, along with the temporal slice up to the current time, denoted as C_t. TranAD functions in two phases. In phase one, the complete temporal sequence C_t is concatenated to a focus score matrix F (originally a zero matrix of length K), and the encoder aims to encapsulate the temporal dynamics of the full sequence through multi-head attention, yielding an encoded representation

I_1. A secondary encoder, termed the window encoder, ingests the input window W_t. It first applies masked multi-head attention to yield an enriched representation I_2, followed by multi-head attention on I_1 and I_2 to generate a context-based representation, analogous to standard transformer architectures. Subsequently, two identical decoders are employed to produce the outputs O_1 and O_2. In phase two, the procedure is repeated, with the focus score recalculated as $F = ||O_1 - W||$, enabling the attention mechanism to prioritize sub-sequences with high deviations. A second output, $\hat{O}_2$, is derived from the second decoder. During the inference phase, for unseen data $(\hat{C}, \hat{W})$, an anomaly score is utilized to identify anomalies, calculated as follows:

$$s = \frac{1}{2}\left\|O_1 - \hat{W}\right\|_2 + \frac{1}{2}\left\|\hat{O}_2 - \hat{W}\right\|_2. \tag{2.24}$$

Yu [162] introduced a structure known as the **Dual Temporal Convolutional Network-Attention** (DTAAD) architecture. This methodology integrates the functionalities of causal and **dilated convolutional networks** (TCN) and a Transformer-based encoder network. The structure is configured to first use two layers of TCN. The first branch, referred to as the local TCN, utilizes causal convolutions to identify local dependencies within a sequence. The second branch, termed the global TCN, employs dilated convolutions to discern global dependencies. Following this, a Transformer encoder is implemented, allowing the model to focus on varied spatial and temporal locations.

2.2.4 Clustering methods

Clustering methods for AD normally rely on enclosing normal data points into one cluster, separating them from anomalous points. These networks are called one-class classifiers.

Classic **one-class classifiers**, such as **One-Class Support Vector Machine** (OCSVM) or **Support Vector Data Description** (SVDD), do not take into account the temporal nature of the input data. Consequently, a feature extractor is often employed when dealing with time-series data. Ergen et al.[163] suggest the joint use of LSTM with either of these two methods, optimizing it in an end-to-end fashion using Quadratic Programming. Furthermore, Shen et al.[164] propose the combination of **Temporal Hierarchical Networks** with **Multiscale Support Vector Data Description** (MSVDD), a variant of SVDD that incorporates multiple layers with multiple hypersphere centers.

Another classic anomaly detection method, called **Isolation Forest** (iForest) uses the Random Forest algorithm to compute an isolation score for each sample of the data. This method has been applied across various tasks, such as intrusion detection, cancer detection, and arrhythmia detection, as discussed by Liu et al. [165]. Numerous adaptations of iForest have been proposed in scientific literature, one of which is the work of Qin et al. [166]. Within the standard iForest algorithm, the anomaly score for each node is derived from

the path length across all iTrees. However, the establishment of the anomaly threshold remains ambiguous. In response, Qin et al. proposed the integration of the K-means algorithm to adaptively segregate anomalous and normal values. A similar approach that fuses K-means with iForest was presented by Karczmarek et al.[167]. Furthermore, Chater et al.[168] introduced the **Deep iForest** (DIF), a method that allows for non-linear partitions in contrast to solely linear isolation. Despite the advancements, these techniques do not take into account the temporal aspect of the data. To address this, recent studies have proposed the combination of iForest with LSTM networks [169, 170, 171].

The **Density-Based Spatial Clustering of Applications with Noise** (DBSCAN) algorithm is a well-regarded method for anomaly detection. Celik et al.[172] applied DBSCAN for identifying atypical patterns in monthly temperature data. Given that DBSCAN does not consider the temporal component of the data, they proposed the removal of the seasonal aspect from the time-series data by implementing the z-score technique as outlined in[173]. Similarly, Wibisono et al. [174] leveraged the DBSCAN algorithm to identify uncommon weather anomalies through the examination of multiple variables.

Anomaly detection can also be achieved by considering information from proximal instances to ascertain the anomalous nature of a point or a set of points. Within this context, the **k-Nearest Neighbourhood** (k-NN) method [175] is commonly used. Here, the closest k instances to each sample in the dataset are computed and an anomaly score based on this distance is derived. The score calculation could either be accomplished by determining the distance d from a sample to its k-th nearest neighbor [176] or by averaging the distance to its k closest neighbors [177]. Subsequently, samples are ranked based on this computed distance d, and a predetermined threshold τ is used to declare a sample as anomalous. In alternative approaches like the **Local Outlier Factor** (LOF) [178], the k nearest instances of each entry are used to calculate a local density and consequently compute a score depicting the ratio of local densities. For a more accurate estimation of the densities of a point p, its reverse nearest neighbors, or those instances that include p in their k nearest neighbors, are also considered [179]. For time-series data, the concept of neighborhood is more intricate due to the ordered nature of the data. Consequently, some works have applied these methods within a sliding window context for a more accurate representation [180, 181, 182].

2.3 Anomaly diagnosis

For a deeper understanding of anomalies, it is not just about detecting them; it is about diagnosing the root causes and characteristics behind them. In the realm of anomaly diagnosis, various methodologies have been explored in the literature. Referring to Figure 2.8, we have categorized the literature into five primary areas: generative networks, attribution-based methods, rule-based methods, counterfactual explanations, and attention-based explanations. This

section will delve into the workings of each approach, guiding us through the different mechanisms employed to interpret and diagnose anomalies.

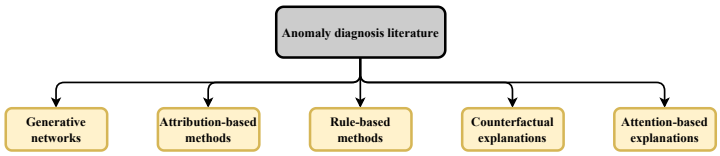


Fig. 2.8: Taxonomy of anomaly diagnosis literature.

2.3.1 Generative networks

Generative networks are neural networks designed to generate new data samples that resemble a given set of training samples.

Among these, **Variational Autoencoders** (VAEs) or **Generative Adversarial Networks** (GANs) have proven to be effective for anomaly detection and interpretation. Ikeda et al.[183], proposed training a VAE in a semi-supervised manner, only using normal data for training. Then, the negative of the Evidence Lower Bound (ELBO) is used as the anomaly score. If a data sample is close to the normal distribution learned by the VAE, the ELBO tends to be higher and vice versa. For interpretation, the algorithm explores the latent distribution of normal data and estimates the contributing dimensions for the anomaly and its degree based on the log-likelihood computed by the latent distribution.

In [184], a **hybrid model consisting of a VAE and One-Class Support Vector Machines** (OCSVM) for anomaly detection is interpreted using Local Interpretable Model-Agnostic Explanations (LIME). Here, the VAE is used for feature extraction and these features are fed to the OCSVM for detecting anomalies and distinguishing between two different faults and unknown factors. The results are then interpreted with LIME, which helps the engineer to make a judgment by prioritizing a set of sensor data that affect the model's decision in detecting abnormality.

As previously mentioned, Su et al. [185] presented OmniAnomaly, an anomaly detection model based on a VAE with Gated Recurrent Units (GRU). They also make an effort to interpret the results obtained in addition to presenting a novel approach for detecting anomalies in multivariate time-series data. The **interpretation is based on the reconstruction probabilities of the time-series**, which are used to estimate the contribution that each dimension had on a detected anomaly. Considering that at time t, the observations $\mathbf{x_t}$ are considered anomalous, $p_\theta\left(\mathbf{x}_\mathrm{t}|\mathbf{z_{t-T:t}}\right) \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\sigma}^2_{\mathbf{x}_t}\mathbf{I}\right)$, an thus, $p_\theta\left(\mathbf{x_t}|\mathbf{z_{t-T:t}}\right) = \prod_{i=1}^{M} p_\theta\left(x_t^i|\mathbf{z_{t-T:t}}\right)$. The purpose is to factorize the conditional probability, $\log(p_\theta(\mathbf{x_t}|\mathbf{z_{t-T:t}}))$, as

$$\log\left(p_\theta\left(\mathbf{x_t}|\mathbf{z_{t-T:t}}\right)\right) = \sum_{i=1}^{M} \log(p_\theta(x_t^i|\mathbf{z_{t-T:t}})) \tag{2.25}$$

in order to estimate the contribution (i.e., the reconstruction probability) of each dimension. Therefore, for each time t, the contribution of each $i \in \{1, ..., M\}$ dimensions is given by $S_t^i = \log\left(p_\theta\left(x_t^i|\mathrm{z_{t-T:t}}\right)\right.$.

Zhou et al. [186] propose Sparse GAN. Here, in addition to presenting an effective method for retinal disease detection on tomography (OCT) images, **explanations of model decisions are given visualizing lesions with Anomaly Activation Maps** (AAM). AAMs are a variation of CAMs in which GAPs are applied over the latent features ($\mathbf{H}_{in}$) of the generator's encoder and over the latent features ($\mathbf{H}_{re}$) of the reconstruction encoder. Then, a vector weight $\mathbf{W}_{aam}$, obtained by $\mathbf{W}_{aam} = \|GAP\left(\mathbf{H}_{in}\right) - GAP\left(\mathbf{H}_{re}\right)\|_1$ is multiplied with $\mathbf{H}_{in}$, obtaining the anomaly activation map.

Chen et al. [156], in their GAN-based model **DAEMON**, compute the anomaly score as the sum of the reconstruction error of each dimension of a given observation x_t, i.e.

$$S_{x_t} = \|x_t - x_t'\|_1 = \sum_{j=1}^{M} \left|x_t^j - x_t'^j\right| = \sum_{j=1}^{M} S_{x_t}^j \tag{2.26}$$

where M is the number of dimension, and $S_{x_t}^j$ is the reconstruction error of the j-th dimension. For anomaly diagnosis, they order the scores $S_{x_t}^j$ in descending order to identify the top-k anomalous dimensions.

2.3.2 Attribution-based methods

Attribution-based methods provide insights into which specific input features contribute most to a model's decision, making them highly valuable for anomaly diagnosis. By highlighting significant features, these methods allow practitioners to identify potential causes of anomalies.

SHAP has been demonstrated to be effective in supervised scenarios, but its effectiveness in unsupervised or semi-supervised scenarios has not been widely validated. Antwarg et al. [187] propose a methodology for using SHAP in unsupervised scenarios. In this work, an Autoencoder (AE) is used to reconstruct the input data, and anomalies are detected by inspecting the reconstruction error. Based on the interquartile range of reconstruction errors, a list of top anomalies is given to 10 field engineers. A visual explanation is then given to the engineers by calculating the negative and positive SHAP values that contribute to and offset the anomaly, respectively. These SHAP values are computed for the top M features that have large reconstruction errors. The field engineers are instructed to decide, using both their systems and the visual depiction provided by SHAP, whether the

anomaly should be further inspected. Takeishi et al. [188] propose a method for computing Shapley values of reconstruction errors of Principal Component Analysis (PCA). The probabilistic view of PCA is used to exactly compute a value function for the Shapley values. In this paper, anomaly detection is done using reconstruction errors obtained with probabilistic PCA. The main problem with this approach lies in how to define the value function v for Shapley values. The authors propose two value functions, v_1 and v_2. While v_1 is inefficient because it requires training PCA for each subset S, v_2 is defined by performing marginalization with respect to unused features to avoid retraining. v_2 is defined as the expected value of the reconstruction error over $p(\mathbf{x}S^c|\mathbf{x}S)$, where $\mathbf{x}S$ denotes the subvector of $\mathbf{x}$ corresponding to the indices in S and $\mathbf{x}S^c$ denotes the complement of S. In this way, Shapley values are computed without the independence assumption using probabilistic PCA.

Giurgiu et al. [189] present another method that utilizes SHAP for detecting anomalies in multivariate time-series data. In this approach, SHAP is used to provide explanations for anomalies detected by a Gated Recurrent Unit (GRU)-based Autoencoder (AE) in an unsupervised manner. Influence weighting is employed to generate informative neighborhoods of samples for computing SHAP values for each signal in a time-series sample. Based on the reconstructed values, the signals are categorized into negative SHAP values, indicating contributing signals when the reconstructed value x_i' is higher than the original value x_i, and positive SHAP values, indicating counteracting signals. The opposite holds true when $x_i < x_i'$. By leveraging SHAP values, lists of the most contributing and counteracting signals for anomalies can be generated.

Rehse et al. [190] describe the process of generating explanations in the DFKI-Smart-Lego-Factory prediction system. They adopt a previous process prediction approach [191] that enables the system to (1) predict the next events and associated resources given an incomplete process instance, (2) estimate the remaining time required for completing the current process step or the entire process, and (3) predict the likely process outcome. For explaining the outcomes, the authors provide various types of explanations, including global feature importance, local rule-based explanations, local feature contribution, and textual explanations. Additionally, feature importance methods in anomaly detection (AD) problems can facilitate root cause analysis, as demonstrated in [192]. Carletti et al. propose an attribution-based method called **DIFFI to interpret the outcomes generated by the Isolation Forest** (IF) algorithm in AD. The DIFFI algorithm operates in unsupervised scenarios and requires access to the structure of the IF model. In their study [192], the algorithm is tested on both a synthetic dataset and an industrial dataset that consists of pressure profiles collected during the vacuum creation process in refrigerator manufacturing.

2.3.3 Rule-based methods

Rule-based methods use predefined conditions and logical constructs to determine outcomes, allowing the users to trace and understand the root causes of identified anomalies.

Regarding rule-based methods, Barbado et al. [193] propose an approach applied to One-Class Support Vector Machines (OCSVM) models for unsupervised outlier detection. Their methodology draws mainly from the work of Martens et al. [194], who introduced an algorithm for extracting rules from an SVM model. In their study [193], the authors extend this idea to OCSVM models by constructing hypercubes that enclose non-anomalous data points. The vertices of these hypercubes are then used as rules to explain when a data point is considered non-anomalous. Furthermore, Kopp et al. [195] employ a rule extraction method, specifically Random Forest, for explaining anomalies. In this approach, a set of trees is trained to provide both minimal and maximal explanations. A minimal explanation comprises a set of rules that contain the minimum number of features necessary to distinguish an anomaly from the rest of the data, while a maximal explanation consists of a set of rules that include all the features in which the anomaly differs from the rest of the data.

2.3.4 Counterfactual explanations

Counterfactual explanations offer insights by presenting scenarios in which an observed outcome would have been different. For anomaly diagnosis, these explanations highlight conditions that help in understanding the specific factors leading to the anomaly.

The diagnosis of anomalies through counterfactual explanations is an emerging area. Trifunov et al. [196] introduce a method using the **Maximally Divergent Interval** (MDI) algorithm to detect anomalous intervals, I, in a time-series, X. In-distribution values substitute anomalous values, $x_i \in I$, which are then reevaluated using MDI. If the substituted interval is less anomalous, the authors conclude that the variables causing the anomaly have been adjusted, thereby considering this substitution as a counterfactual explanation.

The model-agnostic method, Native Guide, proposed by Delaney et al. [197], changes the discriminative regions of a query time-series, T_q, from class c to a different class, c'. This is achieved by minimizing the distance between T_q and its modified version T' while ensuring a different predicted class for T'.

The **CFDet** model [198] is recently introduced to elucidate the decisions of a Support Vector Data Description (SVDD). In this model, an LSTM network, $f(\cdot)$, maps a given set of normal sequences, $\mathcal{P} = \{S_N^+\}_{n=1}^N$, into representation vectors $\mathbf{r}_n^+ = f(S_n^+)$. Subsequently, SVDD establishes a hypersphere enclosing the normal data with the center, $\mathbf{c}$, defined as $\mathbf{c} = Mean(\mathbf{r}_n^+)$. The SVDD is trained to minimize the function $\mathcal{L}_{SVDD}$, with θ being the LSTM parameters:

$$\mathcal{L}_{SVDD} = \frac{1}{N}\sum_{n=1}^{N} \left| f\left(\mathbf{r}_n^+; \Theta\right) - \mathbf{c} \right|_2^2 + \lambda |\Theta|_F^2 \tag{2.27}$$

In this model, an unlabeled dataset $\mathcal{U}$ is first processed to detect anomalous sequences $\tilde{\mathcal{U}}^-$. Following detection, a second LSTM network $g(\cdot)$ is trained using Reinforcement Learning to identify the anomalous entries within an anomalous sequence $Z \in \tilde{\mathcal{U}}^-$.

Guiding the counterfactual generation process using class representative prototypes is a common strategy [199, 200]. Filali et al. [201] propose the use of prototypes via **Dynamic Barycenter Averaging** (DBA) to find centroids for each class. They employ a feature attribution strategy to create saliency maps and perturb the most significant segments.

Generating only a unique counterfactual might not be sufficient, and thus, a variant of **DICE** [202] is proposed by Sulem et al. [203]. In this work, they propose two algorithms: (a) **Interpretable Counterfactual Ensembles (ICEs)** and (b) **Dynamically Perturbed Ensemble (DPE)**, and also their sparse variant to work with high dimensional data. Both algorithms work with multivariate time-series data and produce counterfactual ensembles. ICE is an end-to-end method trained with gradient descent to minimize a loss function that ensures some desired properties, whereas DPE uses an explicit dynamic perturbation method [204].

A recent study by Schemmer et al. [205] states that **counterfactual explanations derived from AEs**, enhance comprehension of the underlying causes of anomalies. Todo et al. [206] use the generative part of a VAE for changing the class-based features of the latent space to produce counterfactual explanations that help to see the differences between healthy ECG signals and ECG signals with pathological patterns. Recently, in [207] AEs have been employed for real-time anomaly diagnosis. Specifically, the research elucidates how the signals from an accelerometer, mounted on a gearbox, alter when the inner race is affected by a crack. In a recent work by Xiao et al. [208], they propose using a generative AE, that uses counterfactual subgraphs that capture the causal relations across different environments,

2.3.5 Attention-based explanations

Attention-based explanations leverage the attention mechanisms in neural networks to highlight important features or regions in the input data. In the context of anomaly diagnosis, these methods shed light on areas within the data that the model pays most attention to when detecting an anomaly.

Attention-based explanations are also used for anomaly diagnosis, as they are typically applied in recurrent networks or in transformers, architectures widely used in anomaly detection due to their ability to process time-series

data. In the domain of anomaly detection within system logs, RNNs, particularly LSTM networks, have demonstrated effectiveness [29]. The LSTM-based model, **DeepLog**, as presented in the aforementioned reference, is designed specifically for anomaly detection and diagnosis from system logs. This research serves as a foundation for subsequent studies, such as the four distinct LSTM-based models proposed for cyber anomaly detection in [30]. Building upon this, Brown et al. [31] incorporated various types of attention mechanisms over these models, thereby achieving state-of-the-art performance whilst providing insights into feature importance and their relational mapping. In a distinct study by Giurgiu et al. [32], an approach for the detection of anomalous events in storage environments was proposed. In this approach, Key Performance Indicators (KPIs) collected from storage environments serve as the primary data source. Expert-defined threshold rules facilitate the extraction of many anomalous events with the aim of predicting storage failures within a three-day window after each 14-day interval. The dataset, denoted as D, is a collection from d storage devices where a set of M metrics (KPI) are collected as time series at regular intervals over a period t. Anomalous events are clustered for each device D_i into chronologically ordered windows $W_{D_i} = \{W_{1D_i}, \ldots, W_{pD_i}\}$. These windows are treated as unordered event sets. For successful future failure prediction, understanding the contribution of each event is crucial. This is achieved through the use of an attention mechanism and a contribution function (dependent on event occurrence time) to represent each window. These window representations serve as the inputs to an LSTM model, which predicts whether a critical event will occur within the interval $[t, t + T]$. The attention weights and contribution function then quantify the contribution of each anomalous event within a window to the decision-making process.

In multivariate time series scenarios, Schockaert et al. [33] presented an approach employing an **attention mechanism and guided backpropagation for generating spatio-temporal explanations**. This study features a regression model predicting the temperature of hot metal produced by a blast furnace. The model comprises a combination of 1D Convolutional Neural Networks (CNNs) and an LSTM. Initially, the multivariate time series are processed through a 1D CNN layer, and the output is concatenated with the original input. The resulting concatenation is then fed into an LSTM layer, generating a unique hidden state $\mathbf{h}_i$ for each time step. A Luong's attention mechanism [209] is utilized on top of the LSTM layer, which generates dynamic attention weights. These weights can be interpreted as the contribution of each time step towards the final prediction. For multivariate time series, spatial information is critical, indicating which dimension of the time series contributes the most to the final decision. In this context, a backpropagation-based approach is employed for each time step t between the attention-adjusted hidden state $\mathbf{h}_t^a$ and the input vector $\mathbf{x}$, highlighting the dimension of the input inducing the gradient in the hidden state $\mathbf{h}_t^a$. This methodology was validated on an artificial dataset and an industrial use case, predicting the temperature

of hot metal produced by a blast furnace. Recently, Ding et al. [210] propose a **multimodal spatio-temporal graph attention network** (MST-GAT) that captures the spatial correlation between different sensors and also the temporal dependencies. Capturing these dependencies facilitates the diagnosis of detected anomalies by highlighting the sensors responsible for the anomaly occurrence.

2.3.6 Transformers

In addition to anomaly detection, transformer-based models have proven to be effective for diagnostics as well, since the multi-head attention used in these networks helps to identify critical points where the model pays attention.

As previously mentioned, **TranAD** [161] uses an architecture that combines two encoder-decoder pairs with multi-head attention, along with a focus score that encapsulates the temporal dynamics of the entire sequence. Special emphasis is given to the task of anomaly diagnosis in their work.

On one hand, they assess the anomaly diagnosis based on the detected anomalies across all sensors. They utilize two metrics for this purpose. The first metric, HitRate@P%, evaluates the number of ground truth dimensions present among the top model predictions. The second metric, Normalized Discounted Cumulative Gain (NDCG) [211], traditionally used in recommender systems to rank the relevance of top listed products, is applied here to rank the anomaly scores obtained for each spatial dimension.

On the other hand, they determine anomaly diagnosis based on the focus and attention scores. This is achieved by visualizing the series along these values to identify where the model's focus lies. Overall, it is observed that the focus and attention scores tend to be higher in regions where anomalies are present.

The strategy followed in **DTAAD** [162] is similar. They evaluate the diagnosis using the same metrics, HitRate@P% and NDCG, on one side. On the other side, they visualize the local and global attention provided by the transformer encoder. Their findings indicate that anomalies and regions with noisy data tend to receive significant attention.

2.4 Critical Analysis of SOTA

A closer examination of the **state-of-the-art techniques reveals a number of weaknesses** that can be further improved upon for more effective anomaly diagnosis.

The majority of current XAI research concentrates on **generating explanations based on the significance each feature has on the model's output**. This includes techniques like SHAP [60], LIME [57], GradCAM [34],

IG [55], etc. In the domain of time-series anomaly detection (AD), such explanations can pinpoint where an anomaly has occurred and even provide insights into the causes of the anomaly.

However, there is a g**ap in the literature when it comes to applying these promising methods specifically for anomaly diagnosis**. The application of these techniques in a real-world, industrial context is often overlooked. Despite there are some works to apply these methods for anomaly diagnosis, for example, Antwarg et al.[187] proposing a methodology for employing SHAP in unsupervised scenarios, or Giurgiu et al.[189] utilizing influence weighting to generate informative neighborhoods of samples and using SHAP values for the purpose of diagnosis. **However, using SHAP requires computing the Shapley values for different possible outcomes. If there are not enough anomalous instances, it might lead to incorrect interpretations**. Therefore, more research is needed to effectively leverage these methods in diagnosing industrial anomalies. This means not only understanding where and why anomalies occur but also designing strategies to apply these methods in a practical, industry-relevant manner. This gap highlights the need for a focused approach to bridge the chasm between academic exploration of these techniques and their potential industrial applications.

Anomaly detection operates on an action-reaction basis - once an anomaly is detected, immediate attention is needed to rectify the issue. In this context, we postulate that **counterfactual explanations could be ideal for anomaly diagnosis**. They can not only help in identifying the problem but also provide guidance for the ensuing actions of the operators, making them a crucial tool in both detecting and effectively responding to anomalies.

Despite their potential, the application of counterfactual explanations to time-series anomaly detection comes with certain limitations. Primarily, **the majority of counterfactual explanation methods are both time-consuming and computationally demanding**, which hinders their efficiency and scalability, particularly in real-time applications. Additionally, **there is a noticeable shortage of methods that have been specifically developed for time-series data**, posing a significant challenge for the application of counterfactual explanations in anomaly detection scenarios. These limitations highlight the need for more tailored and efficient counterfactual explanation methods for time-series anomaly detection.

On another front, we believe **integrating interpretability into anomaly detectors could provide enhanced explanations for diagnosis, as the model is specifically trained for anomaly detection**. Transformers, with their inherent attention mechanisms, may be particularly useful in this context.

However, the application of such methods is not without its challenges. Firstly, **the use of transformer architectures in the domain of XAI remains significantly unexplored**, which is surprising considering their proven success with sequential data. Additionally, **a large proportion of the existing works using transformers resort to basic positional en-**

coding [212, 158], **which is not optimal for identifying high-frequency patterns in data, a crucial aspect for effective anomaly detection in time-series data.**

The application of transformers for anomaly diagnosis is still an emerging area, with a limited number of studies delving into it. Those that do touch upon it tend to use traditional evaluation metrics or attention visualizations, sometimes leaving room for more in-depth insights about the anomalies. This represents a significant gap in the current literature, emphasizing the need for future work to address these shortcomings and leverage transformers more effectively for anomaly diagnosis.

These weaknesses point to potential directions for future research in XAI for anomaly diagnosis. **Improved efficiency, better handling of time-series data, more optimized use of transformer architectures, and more comprehensive methodologies for anomaly diagnosis** are among the improvements that can lead to advancements in the state-of-the-art.

3

Hypothesis, objectives and contributions

In this chapter, we present the hypothesis and the objectives, and detail the contributions of the thesis.

3.1 Hypothesis and objectives

In this study, we aim to explore the potential of XAI approaches for enhancing the identification and understanding of anomalies in time-series data. In order to guide our investigation, we propose two main hypotheses and establish several objectives.

3.1.1 Hypothesis

- **General hypothesis (GH):** In time-series anomaly detection, explainable artificial intelligence methods can provide valuable insights into the decision-making process of black box models, which can lead to proper anomaly diagnosis.
- **Hypothesis 1 (H1):** Post-hoc XAI methods enable real-time interpretation of time series data, providing anomaly diagnosis.
- **Hypothesis 2 (H2):** The integration of interpretability within a black box model can improve the explainability without compromising performance metrics.

3.1.2 Objectives

To address the hypotheses presented above, we set some objectives.

- **General objective (GO):** Research, design, and validate explainable solutions for anomaly diagnosis in time-series data.
- **Objective 1 (01):** Research, design and validate post-hoc real-time explainability for time-series anomaly detection.

- **Objective 2 (02):** Research, design, and validate the integration of intrinsic interpretability in black-box models for time-series anomaly detection and diagnosis.

3.2 Contributions

In this section, we enumerate the contributions of the thesis that are linked to fulfilling the hypotheses and objectives defined in the previous section.

- **Contribution 1 (C1):** We conduct a review of the state-of-the-art and perform a critical analysis to identify the current gaps in the field.
- **Contribution 2 (C2):** We reformulate the Contrastive Explanation Method (CEM) on time series data to generate counterfactual explanations and assess its limitations.
- **Contribution 3 (C3):** We propose a real-time model-agnostic counterfactual explanation method called "Real-Time Guided Counterfactual Explanations (RTGCEx)" and validate it on both image and time-series data.
- **Contribution 4 (C4):** The positional encoding used in existing spatio-temporal transformers has a serious limitation in capturing high frequencies, which tends to ignore mid- and short-range differences in location. To address this, we propose a positional encoding with a mathematical guarantee to alleviate this problem.
- **Contribution 5 (C5):** We propose a supervised architecture based on transformers, capable of learning inherent spatio-temporal relationships that aid in the detection and diagnosis of anomalies.
- **Contribution 6 (C6):** We propose and validate an unsupervised spatio-temporal transformer that uses a masking strategy to extract embeddings from multivariate time-series data based on temporal context. We utilize this architecture for detecting and diagnosing anomalies by measuring the reconstruction differences and attention variations obtained using different length maskings.

  From these contributions, we authored several articles:

- **Article 1 (A1):** Labaien, J., Zugasti, E., & De Carlos, X. (2020, September). *Contrastive explanations for a deep learning model on time-series data.* In Big Data Analytics and Knowledge Discovery: 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings (pp. 235-244). Cham: Springer International Publishing. **Status**: *Accepted.*
- **Article 2 (A2):** Labaien Soto, J., Zugasti Uriguen, E., & De Carlos Garcia, X. (2023). *Real-Time, Model-Agnostic and User-Driven Counterfactual Explanations Using Autoencoders.* Applied Sciences, 13(5), 2912. **Status**: *Accepted.*

- **Article 3 (A3):** Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Diagnostic Spatio Temporal Transfomer with Faithfull Encoding*. Knowledge-Based Systems. **Status**: *Accepted*.
- **Article 4 (A4):** Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Transformers are Efficient Unsupervised Anomaly Detectors*. IEEE Transactions on Pattern Analysis and Machine Intelligence. **Status**: *Under review*.

A diagram illustrating the relationships between the hypotheses, objectives, contributions, and written articles can be found in Figure 3.1.

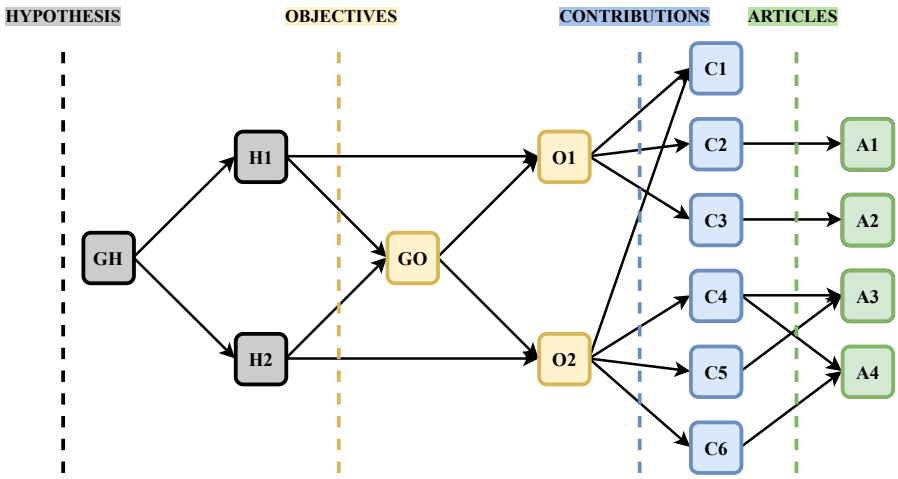


Fig. 3.1: Relation between hypothesis, objectives, contributions, and articles.

Part I

Counterfactual Explanations for Time-Series Data

As we delve into the first part of this thesis, we focus our attention on post-hoc methods for time-series data, an area that holds a significant potential for advancing real-time interpretation and anomaly diagnosis. The study is specifically tailored to address our first hypothesis (**H1**): *Post-hoc XAI methods enable real-time interpretation of time series data, providing anomaly diagnosis.* To pursue this hypothesis, our first objective (**O1**) is to *research, design, and validate post-hoc real-time explainability for time-series anomaly detection.*

Within the broader context of post-hoc XAI methods, we put a particular emphasis on counterfactual explanations, because they not only pinpoint the root of detected anomalies but also provide crucial guidance on subsequent rectification steps. This part of the thesis, therefore, consists of two main contributions, both closely linked to the exploration and enhancement of counterfactual explanations.

The first contribution delves into the use of the Contrastive Explanation Method (CEM) in time-series classification tasks. We uncover both the potential and the limitations of CEM, with a particular focus on its time-consuming nature, which acts as a major hurdle in the path of real-time interpretation.

Motivated by the challenge identified in our initial exploration, the second contribution introduces a novel method, RTGCEx. This real-time, model-agnostic method for generating counterfactual explanations aims to improve upon the time constraints of CEM, offering a more efficient approach.

Together, these contributions form an integral part of our journey to validate our initial hypothesis and fulfill our stated objective, providing a deeper understanding of post-hoc real-time explainability in time-series anomaly detection, specifically through the lens of counterfactual explanations.

4

Contrastive Explanations for a Deep Learning Model on Time-Series Data

In the realm of XAI, a variety of methods, prominently including tools like SHAP, have been employed to explain the decisions made by models. These methodologies are efficient for many tasks, however, when addressing the challenges of detecting and comprehending anomalous patterns within sequential data, a more specialized approach may be necessary.

Our research is particularly motivated by the potential of counterfactual explanations. Such explanations delve into hypothetical scenarios to answer questions of "what if." Not only do they highlight potential areas of concern, but they also provide insights into alternative outcomes, thereby helping decision-makers with guidance on potential remedial actions.

In this first exploration, we are particularly motivated by the Counterfactual Explanations Method (CEM) as outlined by Dhurandhar et al. [71]. Our focus is on the application of this perturbation-based, local explanation technique to time-series data. Although CEM traditionally provides two modes of model interpretation, namely *pertinent negatives (PN)* and *pertinent positives (PP)*, our study is specifically tailored towards the usage of PNs. Unlike other methods, our unique approach is grounded in the application of CEM to time-series data, an area that has yet to be extensively explored.

In this investigation, our hypothesis suggests that employing the Counterfactual Explanations Method (CEM) to time-series data could lead to valuable insights, such as "this time series is classified as class y because a specific point or group of points have a value of v (PP) instead of w (PN)". If this hypothesis holds true, it could indicate that counterfactual explanations serve as powerful instruments for diagnosing anomalies within time-series data. Therefore, our core motivation behind this study lies in dissecting the impact of these explanations when applied to time-series data, navigating any potential difficulties specific to this data type, and ultimately, expanding the utility of counterfactual explanations for anomaly diagnosis.

As CEM needs a model to be explained and an autoencoder to ensure logical perturbations, for the first research of the thesis, we use an LSTM-based classifier, with the incorporated autoencoder also being LSTM-based.

The choice of LSTM networks is motivated by the ability of these models to process time series data. To ensure a comprehensive understanding, the following section provides an in-depth background of the models utilized in this study.

4.1 Background

4.1.1 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks, first proposed by Hochreiter and Schmidhuber [213], are a specific variant of Recurrent Neural Networks (RNNs). Although traditional RNNs have demonstrated their worth in various applications, they struggle with the issues of vanishing and exploding gradients, which hampers their ability to effectively recognize long-term dependencies. LSTMs incorporate unique adjustments to alleviate these difficulties, thus improving their ability to maintain information over extended periods.

A key characteristic of RNNs, and by extension LSTMs, is their chain-like architecture comprised of repeating modules. This design supports the transfer of information over time, allowing each module to relay data to the next. Within the LSTM structure, each of these repeating segments is termed a 'unit'. Every unit consists of four interconnected neural networks that collaboratively manage data processing.

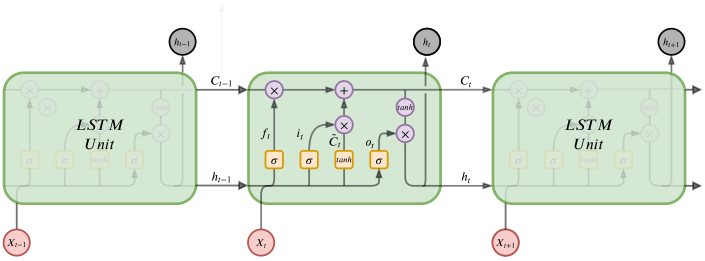


Fig. 4.1: A visualization of an LSTM cell.

A key component contributing to the success of LSTM networks is the cell state ($\mathbf{C}_t$). Serving as the network's memory, the cell state is responsible for transmitting pertinent information throughout the sequence chain. As illustrated in Figure 4.1, the cell state undergoes minimal linear interactions as it traverses the LSTM units, thus aiding the learning of long-term dependencies.

Each LSTM unit also features its own hidden state ($\mathbf{h}$), which supervises the unit's internal condition. The information stored in the cell state is controlled by three gates: the forget gate ($\mathbf{f}_t$), the input gate ($\mathbf{i}_t$), and the output

gate ($\mathbf{o}_t$). The forget gate plays a crucial role in determining what portions of the cell state information are to be retained and what can be discarded. The input gate $\mathbf{i}_t$ identifies which components of the cell state need updating. These updates occur through the addition of new information encapsulated in the candidate vector $\tilde{C}_t$, while simultaneously 'forgetting' elements identified by the forget gate $\mathbf{f}_t$. The final decision involves determining the unit's output, which involves the creation of the output gate $\mathbf{o}_t$ and the hidden state $\mathbf{h}_t$. This intricate procedure is succinctly encapsulated in Equation (4.1).

$$\begin{aligned} f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\ i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{4.1}$$

$\mathbf{W}_x, b_x$ being the weights and the biases of each gate respectively, $x \in \{f, i, C, o\}$.

4.1.2 Autoencoders

Autoencoders (AE) [214] are a type of neural network that is trained in an unsupervised manner with the goal of reconstructing their input data. The architecture of an AE includes two main components: an *encoder* and a *decoder*. Various types of neural networks, such as multilayer perceptrons, convolutional neural networks, and recurrent neural networks, can be employed to construct these components of an autoencoder.

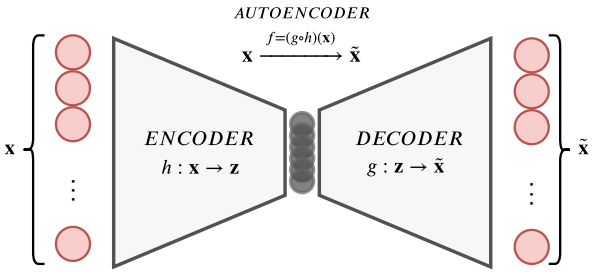


Fig. 4.2: A general structure of an AE.

The structure of an Autoencoder (AE) is illustrated in Figure 4.2. The encoder component can be defined as a function $h : \mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{z} \in \mathbb{R}^d$, where $d < n$. The encoder's role is to compress the data from the input layer

into a lower dimensional latent vector, capturing the most critical features. Conversely, the decoder is a function $g : \mathbf{z} \in \mathbb{R}^d \rightarrow \tilde{\mathbf{x}} \in \mathbb{R}^n$ that takes the compressed latent vector as input and decompresses it into features that closely resemble the original input data. Therefore, an AE can be conceptualized as a function f that maps an input $\mathbf{x} \in \mathbb{R}^n$ to its reconstruction $\tilde{\mathbf{x}} \in \mathbb{R}^n$. It does this by finding the optimal network parameters to minimize a loss function $\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}})$, commonly referred to as the reconstruction loss. This loss is typically quantified using Mean Squared Error (MSE) or Mean Average Error (MAE).

4.2 Methodology

In this study, a model designed for a multiclass time-series classification task is interpreted using Counterfactual Explanations Method (CEM). As outlined in the previous section, the CEM method revolves around solving the optimization problems of Equations (2.7) and (2.9). Consequently, a classification model f must first be proposed, and an Autoencoder (AE) needs to be defined to ensure the modified input remains close to the data manifold. This section presents the proposed classification model and the AE, alongside the description of CEM's application.

4.2.1 Classification model

The proposed model for the classification component combines Long Short-Term Memory (LSTM) with a Fully Connected Layer (FCN). The LSTM processes the data, while the FCN classifies it, as demonstrated by Zhao et al. [215]. Figure 4.3 provides a visual representation of the model architecture.

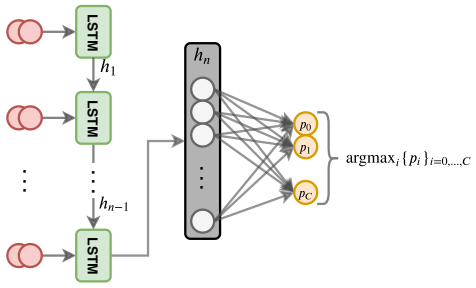


Fig. 4.3: Proposed classification model.

Initially, the model's input is processed by the LSTM layer in time-steps. This procedure allows the LSTM to leverage its previously acquired knowledge in addition to the current input to update the current hidden state. The final

hidden state ($\mathbf{h}_n$) of the network becomes the input for the FCN. The FCN activations $\mathbf{a}$ are computed as:

$$\mathbf{a} = \mathbf{W}_{fc} \cdot \mathbf{h}_n^T + b_{fc} \tag{4.2}$$

where $\mathbf{W}_{fc}$ and b_{fc} are the weights and the biases learned during training process, and $\mathbf{h}_n^T$ is the transpose of the last hidden state. After calculating the activations, a softmax function is applied, which outputs the target as a probability vector $\hat{y}$. Each value $\hat{y}_i \in \hat{y}$ is computed as:

$$\hat{y}_i = \frac{e^{a_i}}{\sum_j e^{a_j}}. \tag{4.3}$$

This value represents the probability of the input belonging to class i. Subsequently, the model takes the index of the maximum probability value as the predicted class. During the training phase, the model's weights are adjusted by minimizing the Categorical Crossentropy Loss (CCE) in batches.

4.2.2 Autoencoders

As outlined in Equations (2.7) and (2.9), the optimization problems include a term associated with an AutoEncoder (AE). This term guarantees that any modifications to the input remain close to the data manifold. The proposed AE architecture relies on Long Short-Term Memory (LSTM) networks and Fully Connected Networks (FCNs), as these structures have proven effective for processing time-series data. The proposed AE is depicted in Figure 4.4.

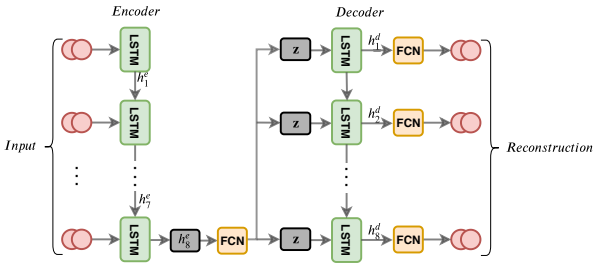


Fig. 4.4: Proposed LSTM-FCN AE.

Initially, the input data is processed using an LSTM network. Subsequently, the final hidden state of the LSTM, which retains information from the entire input, is encoded into a lower-dimensional vector through a Fully Connected Network (FCN). This encoded vector is then replicated to supply the decoder, which is another LSTM. Ultimately, the hidden states of the decoder are connected to an FCN layer. This FCN layer transforms the hidden states

of the decoder into arrays with the same dimensions as the original input, employing a sigmoid activation function. In this setup, the model's weights are adjusted to minimize the Mean Squared Error (MSE).

4.2.3 Contrastive Explanations Method

As stated above, in this work CEM's PNs are studied. In this scenario, the x_0 of Equation (2.7) is a multivariate time-series $\mathbf{x}$, and the δ denotes the PN that makes the predicted class to change (i.e. $\arg\max_i [f(\mathbf{x})]_i \neq \arg\max_i [f(\mathbf{x}+\delta)]_i$), being f the classification model.

Since the changed sample's prediction has to be different from the original class, i.e $\arg\max_i [f(\mathbf{x})]_i \neq \arg\max_i [f(\mathbf{x}+\delta)]_i$, $\max_{i \neq y} [f(\mathbf{x}+\delta)] > [f(\mathbf{x}+\delta)]_y$. Therefore, $[f(\mathbf{x}+\delta)]_y - \max_{i \neq y} [f(\mathbf{x}+\delta)]_i \in [-1, 0)$, and thus, κ has to be chosen in the range $[0, 1]$. In the experiments, a set of different γ and κ parameters have been proved and it is concluded that the best results for this case study are given by $\gamma = 0.2$ and $\kappa = 0.5$.

4.3 Experimental framework

4.3.1 PenDigits dataset description

In this work, a public time-series dataset, named PenDigits, is used. The PenDigits dataset $\mathcal{D}$ is a handwritten digit classification dataset. Each data sample $X_i \in \mathcal{D}$ is a 2-dimensional multivariate time-series, denoted as $X_i = \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}\}$, where $\mathbf{x}_1^{(i)} \in \mathbb{R}^8$ denotes the trajectory of the pen across the coordinate x of a digital screen and $\mathbf{x}_2^{(i)}$ denotes the trajectory of the pen across the coordinate y. Each sample is labeled with a single class label, representing the digit drawn. In Figure 4.5 a sample of the dataset is shown.

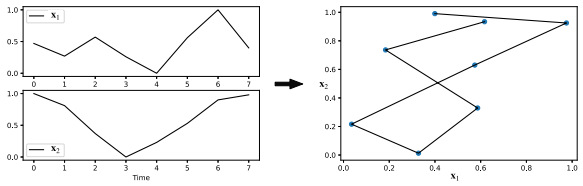


Fig. 4.5: A sample of PenDigits dataset.

The dataset was created by 44 writers and it is divided into two sets: the training set and the testing set. The training test is composed of 7.494 different samples and the testing set is composed of 3.498 different samples.

4.3.2 Framework & hyperparameters

The experimental setup was executed on an Nvidia-Docker container operating on Ubuntu 18.04. The models were constructed utilizing the Keras library within TensorFlow, specifically version 2.1.0. The optimization of both models was conducted using the Adam optimizer. Mini-batches of 32 samples were used for model training, with the process terminating at epoch 163 for the classification model and at epoch 134 for the AE due to the application of EarlyStopping. Initially, the learning rate was set to 0.1, but it was exponentially decayed with each passing epoch. The models were trained using an NVIDIA TITAN V GPU, equipped with 12 GB of memory, on an Intel i7-6850K 3.6Ghz machine supported by 32 GB of DDR4 RAM.

The individual models employed in this research come with their unique set of hyperparameters. The classification model is configured with an LSTM layer comprising 64 units, while the FCN possesses 10 units and utilizes a softmax activation function. The AE features encoder and decoder LSTMs with 16 units each. The FCN within the encoder maps the final hidden state of the initial LSTM into a 4-dimensional vector, thus the encoder's FCN employs 4 units. As the AE is responsible for input data reconstruction, the final FCNs consist of 15 units and use a sigmoid activation function.

4.4 Experimental results of CEM in time-series data

Although the main objective of this work is not to propose a classification model, the model used achieves 98.11% of accuracy and a micro-average F1 score of 0.979 in validation data. On the other hand, the proposed AEs is valid to reconstruct the data, since the MSE for validation data is 0.0064.

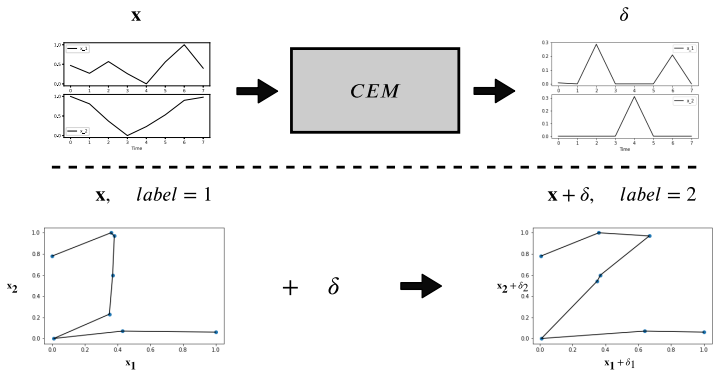


Fig. 4.6: Process used for giving explanations using CEM.

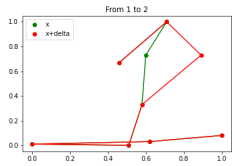


Fig. 4.7: A 1 changed to a 2.

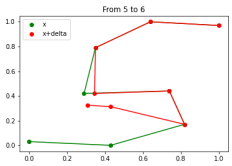


Fig. 4.8: A 5 changed to a 6.

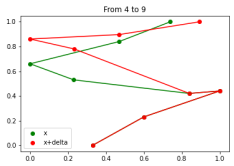


Fig. 4.9: A 4 changed to a 9.

Figure 4.6 illustrates the process used for generating explanations. As depicted in the figure, an input $\mathbf{x}$, which represents the number one, is used to derive its pertinent negatives δ by optimizing Equation 2.7. The pertinent negatives δ represent the modifications necessary to be made to the input $\mathbf{x}$ for the model to classify it as a different class. In this example, it can be observed that by altering the position along the x axis of the third and seventh points and adjusting the position along the y axis of the sixth point, $\mathbf{x}$ is transformed from a one to a two.

In this study, the CEM method was applied to various digit representations, with the explanations provided for three samples illustrated in Figure 4.9. In the first instance, a ”1” is transformed into a ”2” by merely shifting the third point towards the right, emulating the curved shape of the ”2”. In the second case, a ”5” is modified into a ”6” by making three minor adjustments. The most substantial changes occur in the last two points; by shifting them, the CEM creates the round shape at the bottom of a ”6”. In the third scenario, a ”4” is altered into a ”9” by changing the first four points. These changes emulate the rounded shape at the top of a ”9”.

We tested the CEM method on 150 samples, and the adjustments generally make logical sense, leading to a change in the input label. Furthermore, the relationship between the label of the class $\mathbf{x}$ and the label of $\mathbf{x} + \delta$ is consistent across all the samples. Table 4.1 presents the percentages of the changes between the classes of the original and altered samples. It can be seen that, for example, generally the digit “4” is related to the digit “9”, and the changes are done for changing from one to another, and this happens also for all other classes. For example, the digit “2” is converted into a “1” in the 80% of the cases, the changes in digit “7” make it “1” in %54.55 percent of the cases, the digit “8” changes into a “5” in the 50% of the cases, etc.

4.5 Using CEM for time-series data: strengths and weaknesses

With this first research, we have written an article that has been accepted at the 22nd International Conference on Big Data Analytics and Knowledge Discovery, DaWaK 2020:

$\mathbf{x} \rightarrow \mathbf{x}+\delta$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	41.67	0	0	0	50	8.33
1	0	0	25	0	0	12.5	0	25	37.5	0
2	0	80	0	0	0	0	0	0	20	0
3	0	21.43	0	0	0	28.57	0	50	0	0
4	14.29	7.14	0	7.14	0	0	0	14.29	7.14	50
5	0	15.38	15.38	7.7	7.7	0	7.7	0	15.38	30.76
6	23.1	7.7	7.7	0	0	7.7	0	0	53.8	0
7	0	54.55	18.18	0	0	0	0	0	27.27	0
8	11.1	16.67	0	0	0	50	0	0	0	22.22
9	0	16.66	0	0	16.66	16.66	0	0	50	0

Table 4.1: % of changes from $\mathbf{x}$'s class to $\mathbf{x}+\delta$'s class.

Article 1 (A1): Labaien, J., Zugasti, E., & De Carlos, X. (2020, September). *Contrastive explanations for a deep learning model on time-series data.* In Big Data Analytics and Knowledge Discovery: 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings (pp. 235-244). Cham: Springer International Publishing. **Status**: *Accepted*.

This study validates the utilization of CEM in time-series classification tasks, a promising breakthrough given the limited prior application of CEM in such contexts. Results demonstrate that CEM can produce meaningful explanations for these datasets, and it also brings a novel perspective to understanding model decisions via pertinent negatives. Unlike other XAI methods such as SHAP, LIME, LRP, etc., which primarily focus on feature importance, **CEM's pertinent negatives offer insights into what alterations are required in the input to classify it into a different class**. Moreover, the efficacy of CEM extends beyond individual classifications, allowing us to uncover interrelationships between various classes and intuitively comprehend which classes are closely associated.

However, a notable drawback of the CEM approach in its current form is the time-intensive nature of generating explanations. This can potentially hinder its applicability for real-time decision-making, especially in high-dimensional data scenarios. **So, in the next section, we will talk about our next step: creating a new way to make real-time counterfactual explanations that work with any model**.

5

Real-time model agnostic counterfactual explanations

Building on the findings from our exploration of CEM, we turn our attention to the second goal of this thesis. Motivated by the time constraints identified in the CEM methodology, we decided to **develop a method that could provide counterfactual explanations in real-time, adaptable to any model**. This section introduces this agnostic method, designed to offer a quicker yet still effective approach.

In order to start with the development of this algorithm, it is crucial to take into account the specific characteristics that counterfactual explanations should possess to be effectively applicable, particularly in industrial anomaly detection scenarios. It is important to note that these properties may vary depending on the unique context in which they are utilized, as pointed out by Verma et al. [216]. However, certain general properties remain constant across different applications. In this regard, we followed the guidelines provided by [216] regarding the general properties a counterfactual explanation should satisfy:

- **Data closeness:** Given an input $\mathbf{x}$, the counterfactual explanation $\mathbf{x}_{cf}$ has to be a minimal perturbation of $\mathbf{x}$, i.e.,

  $$\mathbf{x} \approx \mathbf{x}_{cf}. \tag{5.1}$$

  Related to data closeness, Verma et al. [216] also mention *sparsity*. However, this property can be a challenge when working with continuous variables because it may not be possible to find a parsimonious explanation by only modifying a few features.
- **User-driven:** The counterfactual explanation has to be user-driven, meaning that the user can indicate whether the class $\mathbf{y}_{cf}$, to which a counterfactual $\mathbf{x}_{cf}$ belongs, can be specified. That is, given a black-box model $f(\cdot)$

  $$f(\mathbf{x}_{cf}) = \mathbf{y}_{cf}. \tag{5.2}$$

  Otherwise, in multi-class problems, for example, the changes would only be directed to the closest class, excluding explanations for other classes.

Note that this property also ensures the *validity* property pointed out by Verma et al. [216].

- **Amortized inference:** The counterfactual explanations have to be straightforward, without solving an optimization problem for each input to be changed. In this way, the explanation model should learn to predict the counterfactual. The algorithm needs to quickly calculate a counterfactual for any new input $\mathbf{x}$. Otherwise, the process of generating explanations is time-consuming.
- **Data manifold closeness:** In addition to satisfying the property of data closeness, a counterfactual instance has to be close to the distribution of the data (p_{data}), i.e., the changes have to be realistic, i.e.,

$$\mathbf{x}_{cf} \sim p_{data}. \tag{5.3}$$

- **Agnosticity:** The generation of counterfactuals can be applied to any machine learning model without relying on any prior knowledge or assumptions about the model.
- **Black-box access**: The generation of counterfactuals can be achieved with access to only the predict function of the black-box model.

Given the required properties for efficient counterfactual explanations, it becomes crucial to optimize a loss function designed to fulfill these properties. This function needs to penalize significant alterations, out-of-distribution counterfactuals, and explanations that don't coincide with the counterfactual label defined by the user. Earlier methods such as CEM [71] and Van Looveren et al. [73] have made notable contributions, yet their reliance on complex optimization problems results in less intuitive explanations. More recently, methods like PIECE [217] and DiCE [202] have emerged, offering more plausible counterfactual explanations. However, they tend to be restricted in their applicability to certain models or binary classification problems.

The constraints of these techniques, particularly their time-intensive nature and complexity of explanations, have spurred the development of *amortized inference methods*. These typically generative models, such as GANs [218, 219] or VAEs [220], are trained to provide instantaneous counterfactual explanations. Despite their advantages, they carry their own limitations like training difficulties, instability, or lower-quality reconstructions.

In response to these challenges, **we propose Real-Time Guided Counterfactual Explanations (RTGCEx), a model-agnostic algorithm designed to generate counterfactual explanations in real-time**. RTGCEx leverages autoencoders to optimize a loss function that embodies all the aforementioned properties.

5.1 Real-Time Guided Counterfactual Explanations

Real-**T**ime **G**uided **C**ounterfactual **Ex**planations (**RTGCEx**) is a model-agnostic algorithm that uses an autoencoder for generating real-time counterfactual explanations for a given black-box model $f(\cdot)$.

In the RTGCEx framework, as depicted in Figure 5.1, the generation of counterfactual explanations is achieved through the interaction of three components: a *Generator (G)*, an *Autoencoder (AE)*, and a *black-box model* $f(\cdot)$. The black-box model $f(\cdot)$ can be any machine learning model, as long as the predict function is accessible. As discussed, the counterfactual explanations produced by RTGCEx must satisfy certain criteria, such as data closeness, closeness to the data distribution, and user-specified explanations, among others. To ensure that these properties are upheld, given a trained black-box model $f(\cdot)$, RTGCEx involves two phases of training:

A. ***Autoencoding phase:***

Involves training an autoencoder by minimizing a loss function that measures the distance between the original samples $\mathbf{x} \in \mathcal{D}$, where $\mathcal{D}$ denotes the dataset on which $f(\cdot)$ has been trained, and their reconstructions $\mathbf{x}'$.

B. ***Counterfactual generation phase:***

This phase involves training G to ensure the properties discussed in the previous section, i.e.,

$$f(\mathbf{x}) \approx \mathbf{x}, \quad f(G(\mathbf{x})) = \mathbf{y}_{cf} \quad \text{and} \quad \mathbf{x}_{cf} \sim \rho_{data}. \tag{5.4}$$

For this purpose, only the weights corresponding to G are trained, but the training makes use of the capacities that $f(\cdot)$ has to classify the inputs and the capacities that AE has to reconstruct the inputs that are similar to the ones learned during the training phase. Training G involves minimizing the following loss function:

$$\begin{aligned} \mathcal{L}^{\mathrm{G},f,\mathrm{AE}}(\mathbf{x}, \mathbf{x}_{cf}, \mathbf{x'}_{cf}\mathbf{y}_{cf}, \mathbf{y'}_{cf}) = & \alpha \cdot \mathcal{L}^{\mathrm{G}}(\mathbf{x}, \mathbf{x}_{cf}) + \\ & \beta \cdot \mathcal{L}^{f}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) + \\ & \gamma \cdot \mathcal{L}^{\mathrm{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf}) \end{aligned} \tag{5.5}$$

where $\alpha, \beta, \gamma > 0$ are regularization coefficients. The first term, $\mathcal{L}^{G}(\mathbf{x}, \mathbf{x}_{cf})$, measures the distance between $\mathbf{x}$ and the counterfactual sample $\mathbf{x}_{cf}$, which ensures that the changes are minimal. The second term, $\mathcal{L}^{f}(\mathbf{y}_{cf}, \mathbf{y'}_{cf})$, measures the distance between the predefined counterfactual class $\mathbf{y}_{cf}$ and the prediction given by $f(\cdot)$ to $\mathbf{x}_{cf}$, i.e., $\mathbf{y}'_{cf}$, which ensures that the input $\mathbf{x}$ is changed to the class $\mathbf{y}_{cf}$ defined by the user. Finally, the third term, $\mathcal{L}^{\mathrm{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf})$, is the reconstruction error of the counterfactual instance $\mathbf{x}_{cf}$ when using the

AE. Because the AE has been trained with instances of the dataset $\mathcal{D}$, the reconstruction error will be smaller when $\mathbf{x}_{cf}$ is similar to the samples used in training, so $\mathcal{L}^{\text{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf})$ penalizes unrealistic changes of the input $\mathbf{x}$, ensuring the *data manifold closeness* property.

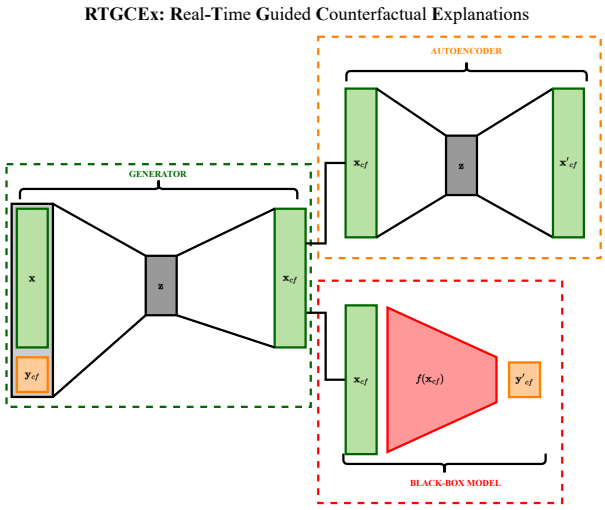


Fig. 5.1: Real-Time Guided Counterfactual Explanations.

5.2 Experimental Framework

In this section, we present a description of the experimental setup. Below, we describe the datasets used, the models employed, and the evaluation metrics.

5.2.1 Datasets

A. ***MNIST.***

The MNIST [221] dataset is a widely-used dataset in the machine learning community, specifically for the task of image classification. It consists of a total of 70,000 images, with 60,000 images being used for training and 10,000 images being used for testing. Each image is a 28 × 28 pixel grayscale image of a handwritten digit, labeled with the corresponding digit (from 0 to 9).

B. ***Gearbox.***

The Gearbox dataset is obtained from a gearbox vibration simulator [222]. The simulator generates vibration data for a gearbox with rotating axes that operate at a fixed speed. The overall operation of the rotatory machine is illustrated in Figure 5.2. The simulator considers an idealized gearbox in which the pinion is coupled to an input shaft connected to the primary engine, while the gear is connected to an output shaft. The shafts are supported by roller bearings located within the gearbox housing. The behavior of the bearings and gearbox housing is monitored using two accelerometers, labeled A1 and A2. In this research, we focus on analyzing the signals obtained from accelerometer A1, which records the contributions of the two shafts, as well as the coupled gear. Depending on the types of failures that can occur within the gearbox, these contributions can vary. In particular, we focus on faults caused by a crack in the inner race of the bearing (as shown in Figure 5.3). This type of fault generates high-frequency vibrations in the gearbox structure between the bearing and the response transducer.

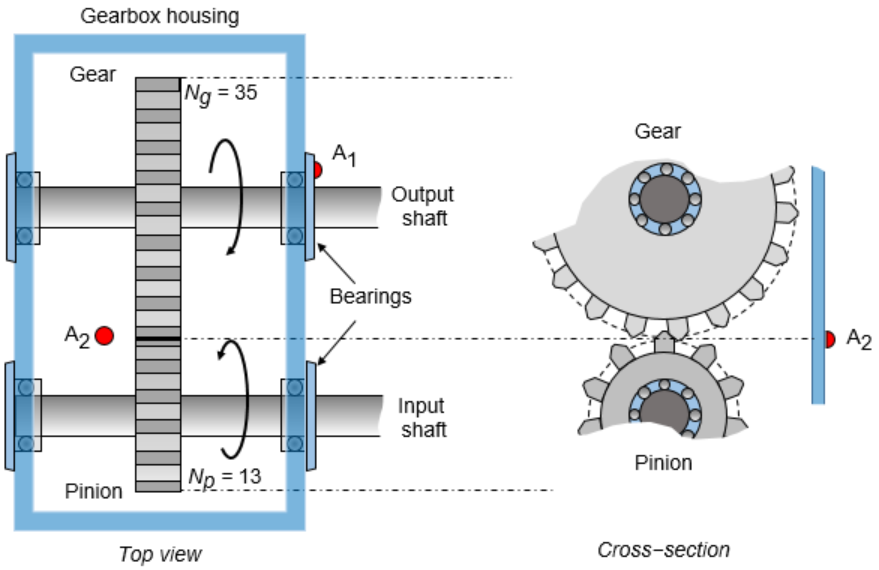


Fig. 5.2: General scheme of the gearbox.

5.2.2 Employed Models

As previously stated, RTGCEx consists of various sub-models. Here we present the architectures used for each use case for these sub-models.

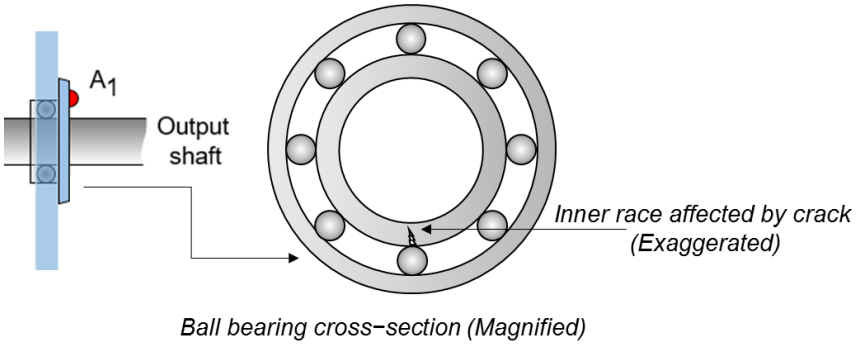


Fig. 5.3: Inner race affected by a crack.

5.2.2.1 MNIST

*A. **Black-box model.***

The black-box model $f(\cdot)$ used has the same architecture as the one used by Van Looveren et al. [73]. That is, the model consists of two convolutional layers with 32 and 64 2×2 filters, respectively, with ReLU activation function. In addition, a 2×2 MaxPooling layer is applied after each convolutional layer to reduce dimensionality and avoid overfitting. Dropout with a fraction of 30% is applied during training. After the second MaxPooling layer, the output is flattened to apply a fully connected layer with 256 units, ReLU activation function, and 50% dropout. Finally, another fully connected layer with 10 units and a softmax activation function is used for classification. For more details, see Figure 5.4.

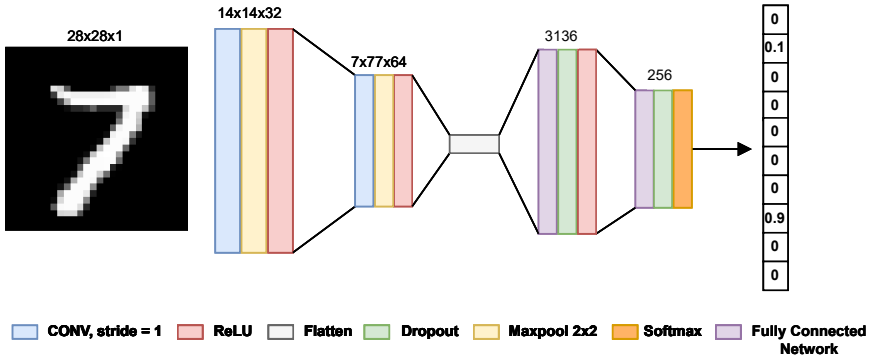


Fig. 5.4: Black-box model $f(\cdot)$ used in MNIST.

B. ***Autoencoder.***

The encoder is composed of two convolutional layers with 32 and 64 2×2 filters, respectively, with ReLU activation function. The output of the second convolutional layer is flattened to feed a fully connected layer with 16 units, from which a 16-dimensional latent vector $\mathbf{z}$ is obtained. Then, this vector is fed to a fully connected layer that transforms the $\mathbf{z}$ vector into a vector of the same dimensionality as the flattened output of the second convolutional layer. This vector is reshaped into a $7 \times 7 \times 64$ tensor to initialize the transposed convolutions. The decoder has three transpose convolution layers, with 64, 32, and one 2×2 filters, respectively. All the transpose convolutions are followed by a sigmoid function. For more details, see Figure 5.5.

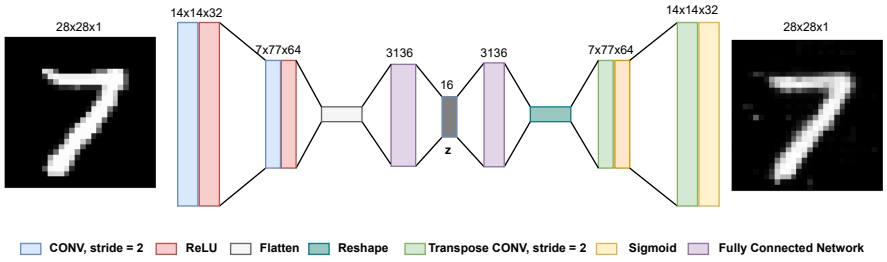


Fig. 5.5: AE used for ensuring data-manifold closeness.

C. ***Generator.***

The architecture of the generator closely resembles that of the AE (see Figure 5.5), except that in this case the latent vector $\mathbf{z}$ is concatenated with the one-hot representation of the class to which the input has to be changed, i.e., $\mathbf{y}_{cf}$.

5.2.2.2 Gearbox

A. ***Black-box model.***

The proposed black-box model $f(\cdot)$ for the classification stage is illustrated in Figure 5.6. This model is based on the anomaly detection model proposed by Cañizo et al. [106], which is a combination of 1-Dimensional Convolutional Neural Networks (1D-CNN) with Recurrent Neural Networks (RNNs). The architecture is designed to be valid for both multivariate and univariate time series. The classificator $f(\cdot)$ is composed of three parts: the CNN backbone, the RNN head, and the classification head.

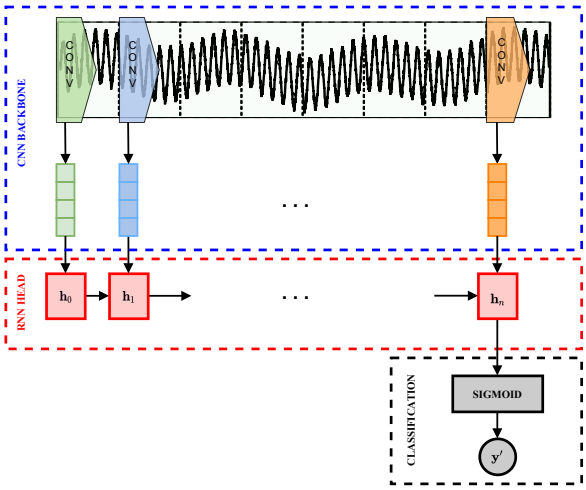


Fig. 5.6: Proposed black-box model.

The CNN backbone, as shown in Figure 5.7, is composed of three convolutional blocks, each containing 1D-CNNs with a ReLU activation function, Batch Normalization, and Dropout regularization layers. The 1D-CNNs in the convolutional blocks are composed of 128, 64, and 32 filters, respectively, with a kernel length of 5 and a stride of 1. The proposed architecture processes the data in a window-based manner, whereby the input data $\mathbf{x} \in \mathbb{R}^L$ are divided into a sequence of T windows $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ and the convolutional blocks are applied to each window separately, resulting in a vector of features $\mathbf{F}_t$ for each time window $\mathbf{x}_t$. These extracted features are the inputs of the RNN head, which is composed of a Bi-LSTM layer that processes the features in chronological order in both directions, allowing for the identification of hidden temporal patterns within the extracted features. Finally, the last hidden state is input into a fully connected layer with a sigmoid activation function, and the final output is computed as

$$\mathbf{y}' = \text{sigmoid}(\mathbf{W}_c \mathbf{h}_T + b_c) \tag{5.6}$$

where $\mathbf{W}_c$ and b_c are the weights and bias of the fully connected layer, respectively.

B. ***Autoencoder.***

The proposed Autoencoder (AE) is composed of a Bi-directional Long Short-Term Memory (Bi-LSTM) network with 128 hidden units for both the encoder and the decoder. The forward and backward hidden states of the encoder are concatenated to obtain the hidden state representation $\mathbf{h}_t^e = [\overrightarrow{\mathbf{h}^e}_t, \overleftarrow{\mathbf{h}^e}_t]$, which contains the context information around time step $t = 1, \ldots, T$. The last

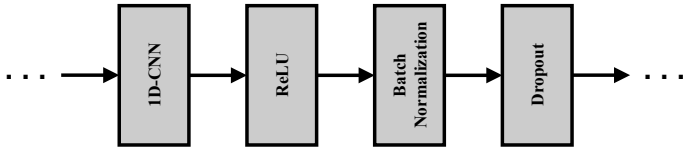


Fig. 5.7: Composition of the convolutional blocks of the CNN backbone.

hidden state $\mathbf{h}_T^e$ is used as the vector representation $\mathbf{z} \in \mathbb{R}^{d_z}$, which contains the compressed information of the input sequence.

In a traditional sequence-to-sequence learning framework, the vector $\mathbf{z}$ is used by the decoder to transform it back into a sequence $\mathbf{x}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_T)$ as close as possible to the encoder input $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$. However, when dealing with long sequences, the latent representation $\mathbf{z}$ may not accurately capture all the information present in the input. To address this issue, attention mechanisms have been introduced to allow the decoder to focus on the most relevant encoder hidden states at each time step. In this work, an attention mechanism is used, whereby the attention weights are computed with an alignment score function. The attention weights at time-step t are calculated as:

$$\boldsymbol{\alpha}_{t,i} = \frac{\exp\left(\text{score}\left(\mathbf{h}_t^d, \mathbf{h}_i^e\right)\right)}{\sum_{j=1}^{T} \exp(\text{score}\left(\mathbf{h}_t^{\text{d}}, \mathbf{h}_j^e\right))}, \tag{5.7}$$

$$\text{s.t} \quad \text{score}\left(\mathbf{h}_t^d, \mathbf{h}_i^e\right) = (\mathbf{h}_t^d)^\top \mathbf{W}_a \mathbf{h}_i^e, \tag{5.8}$$

$$\mathbf{c}_t = \sum_{i=1}^{T} \boldsymbol{\alpha}_{i,t} \mathbf{h}_t^e \tag{5.9}$$

where $\boldsymbol{\alpha}_{t,i}$ are the attention weights, which represent the importance that the input at position i has had over the output at time-step t, $\mathbf{c}_t$ is the context vector, which is the sum of the hidden states weighted by the attention weights, and $\mathbf{W}_a$ is the weight matrix to be learned. The final reconstruction of $\mathbf{x}'_t$ at time-step t is given by:

$$\mathbf{x}'_t = \text{sigmoid}(\mathbf{W}_b[\mathbf{c}_t, \mathbf{h}_t^d]) \tag{5.10}$$

where $\mathbf{W}_b$ is the weight matrix of a fully connected layer.

C. ***Generator.***

For the generator to be able to make minimal changes in $\mathbf{x}$ that result in it being classified as a predefined counterfactual class $\mathbf{y}_{cf}$, the generator must consider both the time series data and the counterfactual class. To accomplish this, the proposed generator in Figure 5.8 is structured similarly to the autoencoder (AE) used for ensuring data manifold closeness, but with the added step of concatenating each input window $\mathbf{x}_t$ with the counterfactual class $\mathbf{y}_{cf}$

so that the information of the counterfactual class is included in each of the hidden states. That is, each input window $\mathbf{x}_t$ now becomes $[\mathbf{x}_t, \mathbf{y}_{cf}]$.

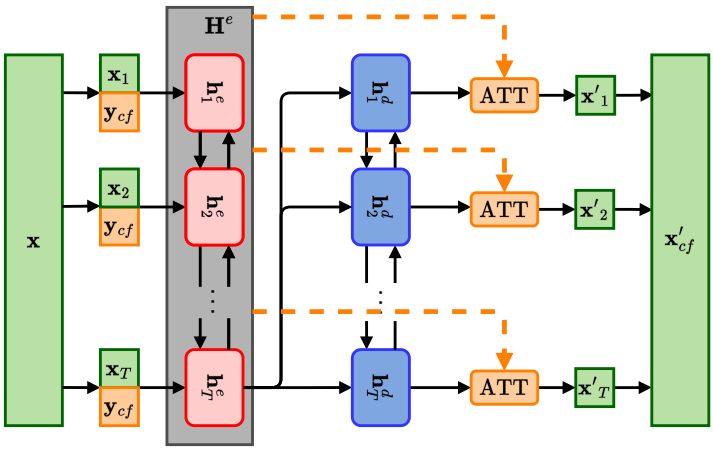


Fig. 5.8: Proposed generator.

5.2.3 Loss Functions

In both use cases, the optimization problem is similar, so the loss functions that we used in both cases are almost the same. As previously described, the training process consists of two phases:

- **Autoencoding phase:** In this stage, we optimize a loss function with the aim of learning to reconstruct the inputs $\mathbf{x}$ from the training dataset. The loss function employed for this purpose is the mean squared error (see Equation (5.11)):

$$\mathcal{L}^{\text{AE}}(\mathbf{x}, \mathbf{x}') = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \mathbf{x}'^{(i)})^2 \tag{5.11}$$

  where N denotes the number of training samples.
- **Counterfactual generation phase:** During this phase, the weights of the autoencoder and the black-box model are maintained constant, and the generator is optimized to minimize a loss function that incorporates three distinct terms (as shown in Equation (5.5)).
  - *Data closeness loss:* This loss function has to minimize the distance between the original samples $\mathbf{x}$ and counterfactuals $\mathbf{x}_{cf}$. Thus, we employ the mean squared error in both use cases:

$$\mathcal{L}^{\mathrm{G}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{cf}^{(i)} - \mathbf{x'}_{cf}^{(i)})^2. \tag{5.12}$$

– *Validity loss:* This loss function has to ensure that the class given by the black-box model y'_{cf} matches the counterfactual class defined by the user, i.e., $f(\mathbf{x}_{cf}) = y_{cf}$. As the MNIST use case is a multiclass problem and the Gearbox use case is a binary classification problem, we use different loss functions here.
On the one hand, for MNIST, we used the categorical cross-entropy loss:

$$\mathcal{L}^{f}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\mathbf{y}_{cf,j}^{(i)} \cdot \log(\mathbf{y'}_{cf,j}^{(i)}) \tag{5.13}$$

where $\mathbf{y}_{cf}^{(i)}$ is the one-hot encoded ground truth label and C is the number of classes.
On the other hand, for the Gearbox use case, we used the binary cross-entropy loss:

$$\mathcal{L}^{f}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_{cf}^{(i)} \cdot \log(\mathbf{y'}_{cf}^{(i)}) + (1 - \mathbf{y}_{cf}^{(i)}) \cdot \log(1 - \mathbf{y'}_{cf}^{(i)})) \tag{5.14}$$

where $\mathbf{y}_{cf}$ is 0 for normal data and 1 for anomalous data.
– *Data manifold closeness loss:* This loss function ensures that the generated counterfactuals are close to the data manifold. Thus, it has to minimize the distance between the generated counterfactuals $\mathbf{x}_{cf}$ and the reconstruction given by the AE for the counterfactuals $\mathbf{x'}_{cf}$. Therefore, the loss used for this was the mean squared error:

$$\mathcal{L}^{\mathrm{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{cf}^{(i)} - \mathbf{x'}_{cf}^{(i)})^2. \tag{5.15}$$

5.2.4 IM1 and IM2 Metrics

To assess the interpretability of the counterfactuals generated with each method, two metrics proposed by Van Looveren et al. [73], IM1 and IM2, have been used. On the one hand, IM1 measures the ratio between the reconstruction error of the counterfactual $\mathbf{x}_{cf}$ using an AE specifically trained with instances of the counterfactual class ($\mathrm{AE}_{\mathbf{y}_{cf}}$) and using another AE trained with instances of the original class ($\mathrm{AE}_{\mathbf{y}}$).

$$\mathrm{IM1}(\mathrm{AE}_{\mathbf{y}_{cf}}, \mathrm{AE}_{\mathbf{y}}, \mathbf{x}_{cf}) := \frac{\left\|\mathbf{x}_{cf} - \mathrm{AE}_{\mathbf{y}_{cf}}(\mathbf{x}_{cf})\right\|_2^2}{\|\mathbf{x}_{cf} - \mathrm{AE}_{\mathbf{y}}(\mathbf{x}_{cf})\|_2^2 + \epsilon} \tag{5.16}$$

A smaller value of IM1 means that the counterfactual instance is easier to reconstruct with the AE trained only with instances of the class $\mathbf{y}_{cf}$ than

with the AE trained only with instances of the original class t_0. This means that the counterfactual instance is closer to the data manifold of the $\mathbf{y}_{cf}$ class instances than to those belonging to the original class t_0. On the other hand, IM2 measures how close the counterfactual instance is to the data manifold. To do so, it compares the reconstructions of the counterfactual instances when using an AE trained with instances of all classes (AE) and the AE trained with instances of the counterfactual class ($\mathrm{AE}_{\mathbf{y}_{cf}}$). To make the metric comparable across all classes, IM2 is scaled with the L_1 norm of $\mathbf{x}_{cf}$.

$$\mathrm{IM2}(\mathrm{AE}_{\mathbf{y}_{cf}}, \mathrm{AE}, \mathbf{x}_{\mathrm{cf}}) := \frac{\left\|\mathrm{AE}_{\mathbf{y}_{cf}}(\mathbf{x}_{cf}) - \mathrm{AE}(\mathbf{x}_{cf})\right\|_2^2}{\|\mathbf{x}_{cf}\|_1 + \epsilon} \tag{5.17}$$

When the reconstruction of $\mathbf{x}_{cf}$ by AE and by $\mathrm{AE}_{\mathbf{y}_{cf}}$ is similar, IM2 is low. A lower value of IM2 makes the counterfactual instance more interpretable, because the data distribution of the counterfactual class $\mathbf{y}_{cf}$ describes $\mathbf{x}_{cf}$ equally well as the distribution over all classes.

5.3 Results of RTGCEx and Discussion

This section analyzes the results obtained with RTGCEx in both the MNIST and Gearbox dataset.

5.3.1 MNIST

In order to train a generator to produce counterfactual explanations, it is necessary to first train both an autoencoder (AE) and a black-box model $f(\cdot)$. The proper functioning of both the AE and the model $f(\cdot)$ is crucial for the generator to operate correctly. As depicted in Table 5.1, the performance of both the autoencoder and the black-box models was evaluated before training the generator. The autoencoder was evaluated using the mean squared error (MSE) on the test set, resulting in a value of 0.007. This suggests that the autoencoder is capable of accurately reconstructing the input data. On the other hand, the black-box model was evaluated using accuracy score, which resulted in a value of 0.991. This implies that the black-box model is able to correctly classify the majority of the images in the test set.

Table 5.1: Results obtained for the AE in terms of MSE and for the black-box model in terms of accuracy.

Model	**AE**	**Black-Box Model**
Metric	MSE	Accuracy
Value	0.007	0.991

After training the generator, experiments were conducted to investigate the performance of the proposed RTGCEx method in comparison with a classical method, Counterfactual Prototype (CFPROTO). Additionally, the impact of the loss function was evaluated by comparing RTGCEx with its variant without the use of the AE. The results of these experiments are illustrated in Figure 5.9, which presents a selection of examples of the generated counterfactual instances. The original instances, along with their corresponding labels, are displayed in the first column, while the counterfactual instances generated by CFPROTO, RTGCEx without AE, and RTGCEx are displayed in the second, third, and fourth columns, respectively. The first row of the figure illustrates how each algorithm converts an original instance of the digit 9 into a 4. In this example, CFPROTO makes minimal modifications to a small part of the upper side of the 9 by removing a few pixels. On the other hand, RTGCEx without AE and RTGCEx make larger changes, but the final result appears more similar to a 4 compared with the counterfactual generated by CFPROTO. Similarly, in the second example, an 8 is converted into a 3, and it is observed that the counterfactual instances generated by RTGCEx without AE and RTGCEx are clearer than the one generated by CFPROTO. In the third example, a 7 is converted to a 9, and all methods produce counterfactual instances that closely resemble a 9, although the instance generated by CFPROTO may be more realistic. Lastly, in the fourth example, a 6 is changed to a 0 by removing the pixels of the lower circle of the 6 and adding new pixels to join the upper part, creating a complete circle.

The effectiveness of the proposed RTGCEx and RTGCEx without AE models were evaluated by generating counterfactual instances for each test sample. In order to ensure a fair comparison, the nearest prototype class $\mathbf{y}_{cf}$ was first generated using CFPROTO, and then instances of the same class were generated using RTGCEx and RTGCEx without AE. The results of the evaluation are presented in Table 5.2, which shows the mean and interquartile range (IQR) of each method in terms of the metrics IM1, IM2, and speed. Following the criteria established for evaluation in [73], we multiplied the IM2 metric by 10.

In terms of IM1, both CFPROTO and RTGCEx performed better than RTGCEx without AE. The results of CFPROTO and RTGCEx were similar, but CFPROTO had less variability, with an IQR of 0.39, compared with 0.58 for RTGCEx. For IM2, RTGCEx outperformed the other two algorithms. When comparing CFPROTO with RTGCEx without AE, it was found that the mean of the values obtained with RTGCEx without AE was 0.08 points lower than that of the values obtained with CFPROTO, and the variability was also lower. The RTGCEx without AE results were significantly improved by introducing a term that penalizes out-of-distribution changes, resulting in a decrease in mean IM2 values from 1.12 to 0.91 and a decrease in variability.

In terms of computational efficiency, CFPROTO took 17.40 s per iteration with a variability of 0.09 s, while RTGCEx without AE and RTGCEx took 0.14 s per iteration with a variability of 0.01 s. It should be noted that RT-

GCEx without AE and RTGCEx have the same architecture and thus take the same amount of time at inference.

Table 5.2: Summary of the results obtained in test for each method. Best results are highlighted in **bold.**

Method	IM1		IM2 (×10)		Speed (s/it)	
	Mean	IQR	Mean	IQR	Mean	IQR
CFPROTO	**1.20**	**0.39**	1.20	0.83	17.40	0.09
RTGCEx wo AE	1.40	0.63	1.12	0.74	**0.14**	**0.01**
RTGCEx	1.21	0.58	**0.91**	**0.63**	**0.14**	**0.01**

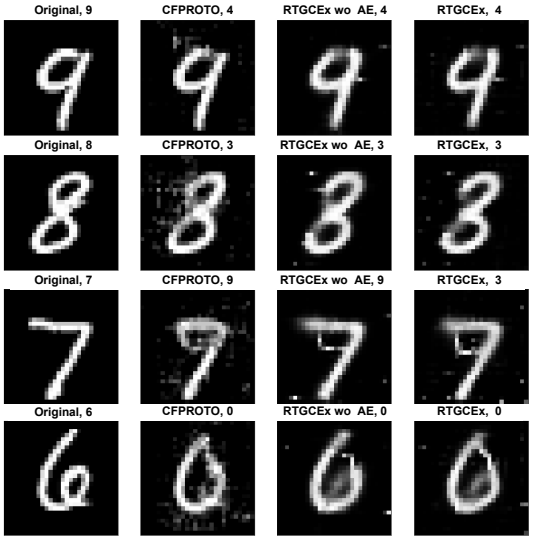


Fig. 5.9: Examples of original and counterfactual instances generated with CFPROTO, RTGCEx without AE, and RTGCEx.

In this research, the comparison of models is conducted by examining the distribution of IM1 and IM2 values, rather than using statistical tests on large samples. This approach is deemed more appropriate because the use of statistical tests on large samples can lead to p-values that are extremely small, which can result in the conclusion of support for results that have little practical significance [223]. We have a test set with 10,000 instances, and the distribution of the IM1 and IM2 values obtained from each method are shown in Figure 5.12. These distributions support the conclusions drawn from

Table 5.2. Specifically, the means of IM1 values for CFPROTO and RTGCEx are similar, while those of RTGCEx without AE and RTGCEx have more variability. Additionally, the mean and variability of IM2 values is lower when using RTGCEx compared with the other methods.

Furthermore, these distributions allow for the drawing of numerical conclusions regarding the comparison of the models, such as the probability that a counterfactual instance will have lower values in terms of IM1 or IM2 when using one method versus another. These comparisons are shown in Table 5.3. For example, the value of 0.66 in the first row and second column of the table indicates that there is a probability of 0.66 that the IM1 value of an instance generated by CFPROTO will be less than an instance generated by RTGCEx without AE. Additionally, we can see that RTGCEx improves drastically over RTGCEx without AE, as an instance generated by RTGCEx has a probability of having lower IM1 and IM2 of about 0.75 compared with one generated with RTGCEx without AE. When comparing RTGCEx with CFPROTO, in terms of IM2, the probability of obtaining better counterfactual instances with RTGCEx is 0.62 with respect to CFPROTO. As for IM1, CFPROTO and RTGCEx have the same probability that the value will be lower using one method or the other.

Table 5.3: This table represents the probabilities that a counterfactual instance has a lower IM1 or IM2 depending on the method used.

Method	**CFPROTO**		**RTGCEx wo AE**		**RTGCEx**	
	IM1	**IM2**	**IM1**	**IM2**	**IM1**	**IM2**
CFPROTO	-		0.66	0.53	0.50	0.38
RTGCEx wo AE	0.34	0.47	-		0.26	0.24
RTGCEx	0.50	0.62	0.74	0.76	-	

Upon analyzing the metrics used to measure the interpretability of counterfactual instances, we have found that IM1 does not always accurately reflect the interpretability of the instances. In some cases, a lower value of IM1 does not necessarily indicate that the counterfactual instances are more interpretable. To illustrate this point, we present two examples of counterfactual instances generated using CFPROTO and RTGCEx in Figure 5.13.

From the visual representation, it can be observed that the instances generated using CFPROTO appear to be less realistic than those generated using RTGCEx. This is reflected in the value of IM2, as instances generated by CFPROTO have a higher value of IM2 than those generated by RTGCEx. However, it is also worth noting that even though the instances generated by CFPROTO are not as realistic, in this case, their IM1 value is significantly lower than that of RTGCEx. In [224], the authors also found that two counterfactuals that were supposed to be similarly realistic exhibited a significant

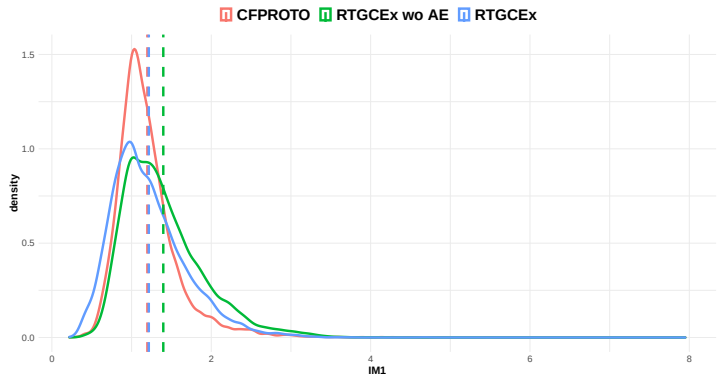


Fig. 5.10: IM1 distribution plots

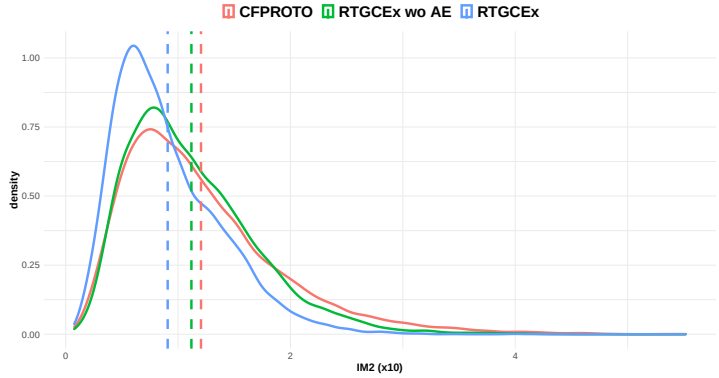


Fig. 5.11: IM2 ($\times 10$) distribution plots

Fig. 5.12: Distribution plots for each method in terms of IM1 and IM2 ($\times 10$) metrics. The dashed lines represent the mean.

difference in the IM1 metric, with one having a notably smaller value than the other. This discrepancy highlights the limitations of using IM1 as the sole metric to measure the interpretability of counterfactual instances.

5.3.2 Gearbox

In order to evaluate the performance of the AE and the function $f(\cdot)$, experimental analysis was conducted. The results of these experiments are presented in Table 5.4.

For the autoencoder, we employed the mean squared error (MSE) metric to evaluate its performance. The low MSE values obtained suggest that the autoencoder is able to accurately reconstruct the input samples. In addition, for the black-box model, we employed three commonly used metrics

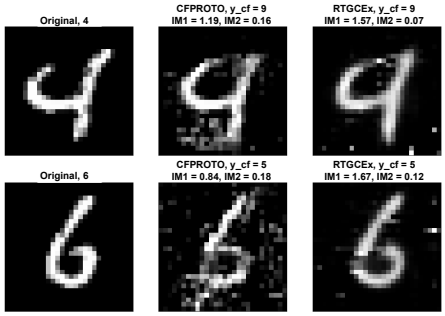


Fig. 5.13: Examples of two counterfactual instances generated by CFPROTO and RTGCEx, where IM2 is lower when using RTGCEx, and IM1 is higher.

in anomaly detection (precision, recall, and F1 score) to evaluate its performance. Our results show that the black-box model was able to effectively detect all anomalies while maintaining a high level of precision.

Table 5.4: Results obtained with the autoencoder and the black-box model $f(\cdot)$.

Model	**AE**	**Black-Box Model**		
	MSE	P	R	F1
Value	2.301×10^{-4}	0.973	1	0.986

In this use case, the generator can be utilized for two distinct purposes: first, it can be employed for identifying and rectifying anomalies by providing the user with an understanding of typical behavior and facilitating the correction of deviances. Secondly, it can be utilized for the generation of anomalous data, by introducing unusual patterns to otherwise standard instances.

Figure 5.14 illustrates the utilization of RTGCEx for addressing anomalies. The input to the model is a simulation that contains anomalies caused by a crack in the inner race. These anomalies occur periodically as the bearing passes through the crack, and they are correctly identified by $f(\cdot)$ with a label of 1. In this scenario, RTGCEx can be employed to determine the minimal modifications necessary to the anomalous simulation for it to be classified as normal by $f(\cdot)$. In the figure, the red signal represents the anomalous journey and the green signal represents the signal generated by RTGCEx. Upon closer examination, it can be observed that RTGCEx focuses on altering the areas where the anomalies appear, as indicated by the red shaded area.

As previously mentioned, in addition to its use for identifying and rectifying anomalies, RTGCEx can also be utilized as a generator of anomalous

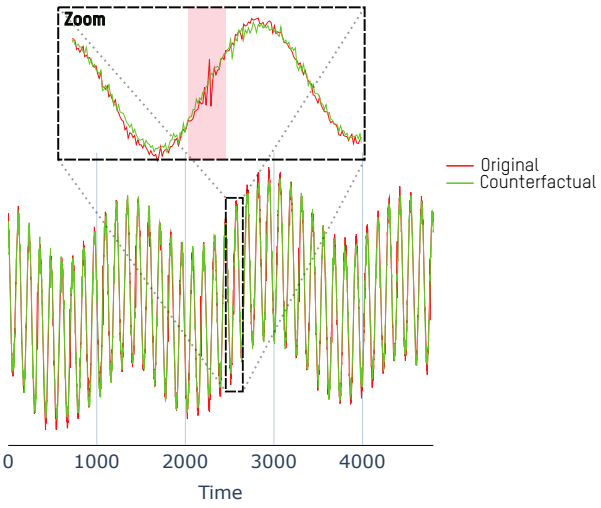


Fig. 5.14: An anomalous sample and the counterfactual explanation for showing normal behavior.

data. Figure 5.15 provides an example of this application. The figure shows an input representing a non-anomalous journey of the gearbox, represented in red, and the changes that would need to be made, represented in green, to create an anomalous journey. It can be observed that RTGCEx has introduced several anomalies in the input data, highlighted by the red shaded areas, simulating the behavior that the accelerometer would have should cracks form in the inner race. This capability allows for the creation of anomalous data for understanding unusual behaviors or to create anomalous data in unbalanced scenarios, for example.

To measure the validity of the generated counterfactuals and the influence that each term of the loss function has on the final result, in Table 5.5, we measured the interpretability based on the three terms of the loss function of Equations (5.12), (5.14) and (5.15). In this table, each row corresponds to the results obtained according to which terms were used in the loss function, i.e.,

$$\begin{aligned}
\textbf{RTGCEx wo } \mathcal{L}^{\text{AE}}: &\quad \mathcal{L} = \alpha \cdot \mathcal{L}^{\text{G}}(\mathbf{x}, \mathbf{x}_{cf}) + \beta \cdot \mathcal{L}^{\text{D}}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) \\
\textbf{RTGCEx wo } \mathcal{L}^{\text{G}}: &\quad \mathcal{L} = \beta \cdot \mathcal{L}^{\text{f}}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) + \gamma \cdot \mathcal{L}^{\text{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf}) \\
\textbf{RTGCEx}: &\quad \mathcal{L} = \alpha \cdot \mathcal{L}^{\text{G}}(\mathbf{x}, \mathbf{x}_{cf}) + \beta \cdot \mathcal{L}^{\text{f}}(\mathbf{y}_{cf}, \mathbf{y'}_{cf}) + \\
&\qquad \gamma \cdot \mathcal{L}^{\text{AE}}(\mathbf{x}_{cf}, \mathbf{x'}_{cf})
\end{aligned} \tag{5.18}$$

This table presents the results of an evaluation of each term's impact on the RTGCEx algorithm's loss function on counterfactual generation. The results demonstrate that the term which ensures data similarity (represented by $\mathcal{L}^{G}$) is crucial for generating counterfactual explanations that are highly similar

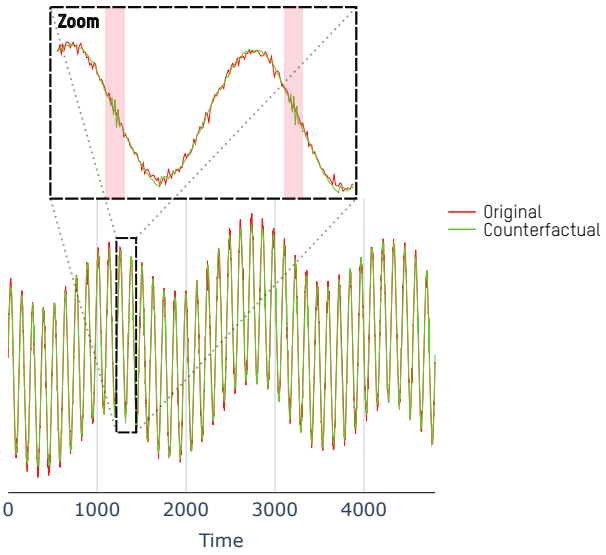


Fig. 5.15: A normal sample and the counterfactual explanation for introducing anomalies.

Table 5.5: Mean values obtained for each loss function in test data using the three variations of the losses optimized in RTGCEx.

Method	Closeness Loss	Counterfactual Loss	Data Manifold Loss	Speed (s) One Sample	Test Data
RTGCEx wo $\mathcal{L}^{\text{AE}}$	3.23×10^{-4}	4.91×10^{-7}	1.60×10^{-4}		
RTGCEx wo $\mathcal{L}^{\text{G}}$	0.056	4.18×10^{-7}	2.32×10^{-4}	0.062 ± 0.091	5.46
RTGCEx	3.59×10^{-4}	5.83×10^{-7}	1.29×10^{-4}		

to the input data and involve minimal changes. Removal of this term from the loss function leads to counterfactuals that are dissimilar to the input data and involve significant changes. The numerical analysis clearly shows this, as the closeness loss increases from 3.59×10^{-4} to 0.056, and the data manifold loss increases from 1.29×10^{-4} to 2.32×10^{-4} upon removal of this term. Inclusion of this term in the loss function results in small values in data closeness loss, specifically 3.23×10^{-4} for RTGCEx without $\mathcal{L}^{\text{AE}}$ and 3.59×10^{-4} for RTGCEx, indicating that the generated counterfactuals involve minor modifications to the input data. Furthermore, the value of the data manifold loss is low for RTGCEx and also when $\mathcal{L}^{\text{AE}}$ is removed, indicating that the changes made are logical in both cases. The counterfactual loss is not significantly different across all methods, as the class of the counterfactual instances matches the intended class in all cases.

The amortized inference requirement is particularly important in anomaly detection [205], as a quick response is often required when an anomaly occurs.

Therefore, the table also shows the average time it takes for the model to generate explanations for a single instance and for the entire test dataset (2020 instances). All methods take the same amount of time, as the generator used to generate the explanations has the same architecture in all three cases. For a single explanation, the generator takes on average 0.062 s with a variation of 0.091 s, while generating explanations for the whole test set takes 5.46 s. This demonstrates that the proposed method provides counterfactual explanations in real time, satisfying the amortized inference property.

5.4 RTGCEx, effective and fast solution for user-driven counterfactual explanations

With this research, we have written an article that has been accepted the journal Applied Sciences of MDPI:

Article 2 (A2): Labaien Soto, J., Zugasti Uriguen, E., & De Carlos Garcia, X. (2023). *Real-Time, Model-Agnostic and User-Driven Counterfactual Explanations Using Autoencoders.* Applied Sciences, 13(5), 2912. **Status**: *Accepted.*

Building on the challenges discovered in the usage of CEM, specifically its time-consuming nature, **this study introduces RTGCEx, a real-time, model-agnostic method for generating user-driven counterfactual explanations**. This proposed method takes advantage of autoencoders trained with a multiobjective loss function to generate valid and data-close counterfactuals that match the user's desired outcome.

RTGCEx has shown its effectiveness across multiple domains and data types, serving as a solution to the time issue associated with CEM. Our experiments, particularly those conducted on the MNIST dataset, reveal that **RTGCEx surpasses traditional methods not only in speed but also in effectiveness**. It achieves lower IM2 values while preserving similar IM1 values in a fraction of the time. It is also worth noting that IM1 does not always accurately represent the interpretability of the generated instances, as instances with lower IM1 values were sometimes less realistic.

Our tests on the Gearbox dataset further demonstrate the robustness of RTGCEx in detecting and rectifying anomalies. It consistently resulted in low values across all terms of the loss function, satisfying the desired properties. **This suggests that RTGCEx stands as a compelling solution to the issues earlier identified in CEM, delivering fast and efficient counterfactual explanations**.

Part II

Contributions to anomaly diagnosis using transformers

In the area of machine learning and its varied applications, there exists a need for models to understand their decisions, especially in critical domains such as anomaly detection. The initial part of this thesis delved into methods offering explanations after the model's decision-making process for real-time interpretation of time-series data. **However, using post-hoc methods might not always capture the nuanced intricacies inherent in the data and models, as these explanations are independent of model internals, and essential information can be lost**. This understanding underscored the potential benefits of **embedding interpretability directly into anomaly detectors to offer enhanced explanations during the diagnosis phase**.

Building upon our earlier exploration in the first part of this thesis, where we examined post-hoc XAI methods for real-time interpretation of time-series data, we now shift our focus to the second part of our thesis. Here, we focus on the creation of inherently explainable models, specifically tailored for anomaly detection. We believe such a development aligns with our second hypothesis (**H2**): *The integration of interpretability within a black box model can improve the explainability without compromising performance metrics.* Guided by this hypothesis, we work towards our second objective (**O2**): to *research, design, and validate the integration of intrinsic interpretability in black-box models for time-series anomaly detection and diagnosis.*

This part of the thesis has two main contributions, both related with the concept of leveraging Transformer models for anomaly detection and their ability to extract spatio-temporal dependencies in an interpretable manner.

Our first contribution addresses the challenge of anomaly diagnosis in multi-sensor data. **We propose a novel framework - the DFSTrans - which uses a Transformer-like architecture to learn spatial and temporal dependencies from sensor data in a supervised manner**. **This study also identifies the limitations of traditional positional encoding and proposes a new method, the *faithful encoding***, that is more effective for anomaly detection tasks.

Given the valuable insights gained, we move towards the exploration of unsupervised approaches, inspired by the need for such methods in anomaly detection, where acquiring labeled anomalies is often a challenging task. Hence, **our second contribution presents an unsupervised version of our proposed DFSTrans framework, termed uDFSTrans. This approach combines a multi-masking strategy and a context-based attention mechanism for the detection and diagnosis of anomalies in multivariate time series data. It offers an efficient unsupervised solution for anomaly detection, allowing us to understand the outputs of such models.**

Collectively, these contributions significantly improve our understanding of how to integrate interpretability directly within anomaly detection models. They validate our initial hypothesis and fulfill our objective, illustrating the

capabilities of Transformer models in enhancing the diagnosis of anomalies in time-series data.

6

Diagnostic Spatio-temporal Transformer with Faithfull Encoding

In this study, our focus is on examining the potential of Transformer models in diagnosing anomalies. We tackle the problem from a supervised learning perspective, which provides us with a structured framework to scrutinize the capabilities of these models in detail.

One of the primary factors contributing to the difficulty of anomaly diagnosis in multi-sensor data is the absence of prior knowledge regarding the spatial and temporal scales that are pertinent to the anomalies of interest. Some failures may be characterized by high-frequency jitters, while others may be due to relatively long-term shifts of certain signals. Also, some failures may be detected as an outlier of a single sensor, while other failures may involve multiple sensors. The key technical challenge in anomaly diagnosis is to automatically learn the higher-order interactions between the temporal and spatial indices inherent in the multi-sensor data [225].

For flexibly capturing various temporal scales and dependencies, Transformer networks look like a promising approach. Recently, *spatio-temporal* (ST) extensions of the Transformer model have attracted attention in a variety of tasks such as human motion tracking and traffic analysis. Unlike conventional convolutional networks [226], Transformer networks have the potential to capture complex ST dependencies in a highly interpretable fashion, making them an attractive choice for anomaly diagnosis. There have been mainly two approaches in ST transformers in the literature: One is the *twin* approach, which builds two transformer networks for the spatial and temporal dimensions [227, 228, 229]. The other is the *hybrid* approach, which typically combines a graph convolutional network (GCN) with a temporal transformer [230, 231, 232].

Although those works report promising results in their targeted applications, there are **three major limitations** in our context:*First*, the **hybrid transformer approach requires prior knowledge of the spatial dependency**, typically in the form of physical distance, which is not available in most industrial anomaly diagnosis settings; *Second*, the **twin approach uses essentially the same sinusoidal positional encoding as proposed**

in [28] without clear justification; and *Third*, their main focus is temporal forecasting, and hence, **they do not provide readily consumable information for anomaly diagnosis**.

In this thesis, we formalize first the anomaly diagnosis problem as supervised dependency discovery, where ST dependencies are learned as a side product of a multivariate time-series classification task (see Sec. 6.1 for the detail). Our framework features a novel temporal encoding approach by combining a new positional encoding algorithm with a 1D multi-head CNN. We mathematically show that conventional sinusoidal encoding has a serious limitation in capturing short-scale variability. As outlined in Fig. 6.2, an embedded time-series episode enriched by the positional encoding is passed to a ST dependency model. Thanks to a proposed ST factorization model, our framework can provide local and global diagnostic scores, which can be readily consumed in real-world anomaly diagnosis tasks. Unlike the existing hybrid ST transformer approaches, it can also learn spatial dependency without prior knowledge.

In summary, with the approach presented in this chapter, we contribute with (1) a new positional encoding algorithm called the DFT (discrete Fourier transform) encoding that has a faithfulness guarantee and (2) an ST diagnostic approach based on the proposed ST dependency discovery model.

6.1 Problem setting

This section states the problem setting and explains the dependency discovery problem.

*A. **Training data***

We are given N sets of S-variate time-series and a binary label as $\mathcal{D}_{\text{train}} \triangleq \{(\mathsf{X}_i, y_i) \mid i = 1, \ldots, N\}$. Each of the S time-series typically corresponds to measurements of one physical sensor. The binary label y_i is 1 if X_i is anomalous and 0 otherwise. Although anomalous samples are generally hard to obtain in many real-world anomaly *detection* settings, we assume that reasonably many positive samples have been collected for the purpose of anomaly *diagnosis*. Each time-series episode is divided into N_w consecutive segments using a sliding window of size w_l as $\mathsf{X}_i = [\mathsf{X}_i^{(1)}, \ldots, \mathsf{X}_i^{(N_w)}]$, where $\mathsf{X}_i^{(t)} \in \mathbb{R}^{S \times w_l}$ is the t-th time-segment. An example of a time-series episode is an ascend or descend session of an elevator. Note that both temporal and spatial dependencies matter within one episode, while different episodes are considered statistically independent. Unlike human motion tracking and traffic monitoring [233, 230] we do not assume prior knowledge of the similarity in the spatial dimension (i.e., among sensors). See Figure 6.1 for a detailed overview of the notation used.

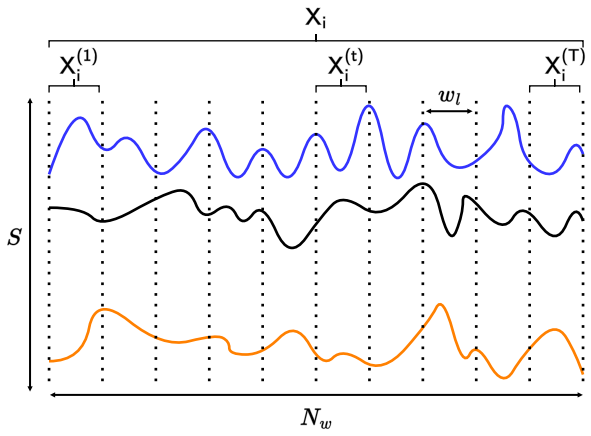


Fig. 6.1: Illustration of the notation used. One episode of S-variate time-series is split into N_w consecutive segments of size w_l. A binary label y_i is associated with the time-series episode X_i, where $y_i = 1$ means that the episode X_i is anomalous.

B. ***Supervised dependency discovery problem***

From a practical perspective, our goal is to obtain actionable insights into anomalous samples by providing explanations of how they are anomalous. Specifically, major questions we wish to answer include (1) at a moment that an unusual situation is observed in one or a few variable(s), how the other variables are related, and (2) in a sensor showing an unusual behavior, how the unusual behavior might have been triggered by past events.

As a machine learning task, this can be done by solving the *supervised dependency discovery* problem. Specifically, we train a binary classifier using a prediction model that incorporates a learnable spatio-temporal (ST) dependency model (explained in Sec. 6.3.3) as part of the data generative process. If the classification accuracy is high enough, the learned dependency can be used as a proxy of real dependency structure. Although our objective function is simply the binary cross-entropy (BCE):

$$\ell[\mathcal{M}] = -\sum_{i=1}^{N}\{y_i \ln p(y_i = 1 \mid \mathsf{X}_i) + (1 - y_i)\ln[1 - p(y_i = 1 \mid_i)\}, \tag{6.1}$$

where $p(y_i = 1 \mid \mathsf{X}_i)$ is the probability of X_i being anomalous (i.e. the output given by the sigmoid function) and $\mathcal{M}$ symbolically represents the ST dependency model, our main motivation is to find $\mathcal{M}$ through maximizing the classification performance.

6.2 Background on Transformers

The Transformer algorithm consists of two steps: Positional encoding (PE) and self-attention filtering. Typically, before implementing these steps, it is common to define a function $\psi(\cdot)$ that maps the raw data space X to an embedded data space F.

$$\psi : \mathsf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \mapsto \mathsf{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \tag{6.2}$$

Positional encoding

The goal of PE is to reflect the information of the position of the items (i.e., words in machine translation) in the input sequence. In the original formulation [28], this was done by simply adding an extra vector to the representation vector of an input word:

$$\boldsymbol{f}_t \leftarrow \boldsymbol{f}_t + \boldsymbol{e}_t, \tag{6.3}$$

where $\boldsymbol{f}_t$ is the representation vector of the t-th item in the sequence and $\boldsymbol{e}_t$ is its corresponding positional encoding (no spatial index s was considered so is omitted). They used the following form for $\boldsymbol{e}_t$:

$$(\sin(w_0 t), \cos(w_0 t), \sin(w_2 t), \cos(w_2 t), \cdots, \sin(w_{d-2} t), \cos(w_{d-2} t))^\top \tag{6.4}$$

with $w_k = \rho^{-\frac{k}{d}}$, $\rho = 10\,000$, and $d = 512$.

Self-attention filtering

With Eq. (6.2), we now have an updated data matrix $\mathsf{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$. The self-attention filtering further updates F to get Z. There are three steps in the algorithm. The first step is to create three different feature vectors from F:

$$\mathsf{Q} \triangleq \mathsf{W}_Q \mathsf{F},\ \mathsf{K} \triangleq \mathsf{W}_K \mathsf{F},\ \mathsf{V} \triangleq \mathsf{W}_V \mathsf{F}, \tag{6.5}$$

which are called the query, key, and value respectively. $\mathsf{W}_Q, \mathsf{W}_K, \mathsf{W}_V$ matrxices are $d \times D$ and have to be learned from data. Considering that $\mathsf{Q}, \mathsf{K}, \mathsf{V}$, as well as F are column-based data matrices, we denote the column vectors as $\mathsf{Q} = [\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N)}]$, $\mathsf{K} = [\boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(N)}]$, and $\mathsf{V} = [\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(N)}]$.

The second step is to compute the self-attention matrix A. From the query and key matrix, the self-attention matrix $\mathsf{A} \in \mathbb{R}^{N \times N}$ is defined as

$$\mathsf{A} \triangleq \mathrm{softmax}\left(\frac{1}{\sqrt{d}} \mathsf{Q}^\top \mathsf{K} + \mathsf{M}\right), \tag{6.6}$$

where the softmax function applies to each row and M is a masking typically applied that varies, which varies depending on the context in which transformers are utilized. For the sake of simplicity, we will omit the masking in the following equations. To write down the (i, j) element explicitly, we have

$$A_{i,j} = \frac{\exp(B_{i,j})}{\sum_{m=1}^{N} \exp(B_{i,m})}, \quad B_{i,j} \triangleq \frac{1}{\sqrt{d}} (\boldsymbol{q}^{(i)})^{\top} \boldsymbol{k}^{(j)}. \tag{6.7}$$

This definition implies that A is essentially a cosine-similarity matrix between the items i and j.

Finally, the third step is to take in the influence of neighbors in each row:

$$\boldsymbol{z}_t = \sum_{k=1}^{N} A_{t,k} \boldsymbol{v}^{(k)} \quad \text{or} \quad \mathsf{Z} = \mathsf{V}\mathsf{A}^{\top} \tag{6.8}$$

Obviously, through this step, the representation vectors are encouraged to have similar values to their neighbors.

Typically, self-attention filtering is multiplexed by using multiple "heads." [1] Let H be the number of heads. In this approach, H different sets of the parameter matrices are used $\{\mathsf{W}_Q^{[h]}, \mathsf{W}_K^{[h]}, \mathsf{W}_V^{[h]} \mid h = 1, \ldots, H\}$ with different initializations. As a result, H different self-attention matrices $\{\mathsf{A}^{[h]}\}$ and value vectors $\{\boldsymbol{v}_t^{[h]} = \mathsf{W}_V^{[h]} \boldsymbol{f}_t\}$ are obtained. For each, the representation vectors are computed as

$$\boldsymbol{z}_t^{[1]} = \sum_{k=1}^{N} A_{t,k}^{[1]} \boldsymbol{v}_k^{[1]}, \quad \ldots, \quad \boldsymbol{z}_t^{[H]} = \sum_{k=1}^{N} A_{t,k}^{[H]} \boldsymbol{v}_k^{[H]} \quad \in \mathbb{R}^d. \tag{6.9}$$

The final representation is created simply by concatenating these H vectors as

$$\boldsymbol{z}_1 = \text{concat}\left(\boldsymbol{z}_1^{[1]}, \ldots, \boldsymbol{z}_1^{[H]}\right), \ldots, \boldsymbol{z}_N = \text{concat}\left(\boldsymbol{z}_N^{[1]}, \ldots, \boldsymbol{z}_N^{[H]}\right) \quad \in \mathbb{R}^{dH}. \tag{6.10}$$

After this, to normalize the output, reduce the internal covariate shift, and introduce non-linearities to capture complex patterns a normalization layer followed by two feedforward layers is usually applied to the enriched representation Z,

$$\mathsf{Z} \leftarrow \text{LayerNorm}\Big(\text{LayerNorm}(\text{Dropout}(\mathsf{Z})) + \tag{6.11}$$

$$\text{Dropout}(\text{ReLU}\left(\mathsf{Z}\mathsf{W}_1 + b_1\right)\mathsf{W}_2 + b_2\Big) \tag{6.12}$$

where $\mathsf{W}_1, \mathsf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ and $b_1, b_2 \in \mathbb{R}^{d_{ff}}$ are the weights and biases of the first and second feedforward layers, respectively.

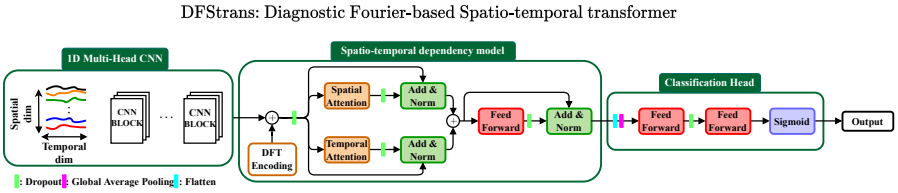


Fig. 6.2: Architecture of the proposed DFStrans.

6.3 Proposed spatio-temporal dependency discovery framework

The proposed anomaly diagnosis framework consists of three major components as shown in Fig. 6.2. This section first states the problem setting and explains the major components in detail.

6.3.1 Overall model architecture

The goal of automatically capturing informative ST patterns requires the capability of handling different temporal resolutions as well as learning ST dependencies.

For the former, we employ a multi-head one-dimensional (1D) convolutional neural network (CNN), similar to the one used by Canizo et al. [234] (see Figure 6.3), which consists of multi-head 1D convolution, MaxPooling, ReLU activation, and batch normalization, applied independently to each of the variables (see Appendix for the detail). This module defines a mapping from raw time-series segment to a representation matrix: $\mathsf{X}_i^{(t)} \rightarrow \mathsf{F}_i^{(t)} \in \mathbb{R}^{S \times M}$, where M is the dimensionality of the representation vector. To highlight spatial-temporal aspects of the data, we mainly use $\{\boldsymbol{f}_i^{(t,s)}\}$ instead of $\mathsf{F}_i^{(t)}$, where $\boldsymbol{f}_i^{(t,s)} \in \mathbb{R}^M$ and t, s run over $1, \ldots, N_w$ and $1, \ldots, S$, respectively. Unless confusion is likely, we omit the sample index i from $\boldsymbol{f}_i^{(t,s)}$ hereafter.

As shown in Fig. 6.2, the output of the 1D CNN module is fed into the transformer module. The feature tensor $\{\boldsymbol{f}^{(t,s)}\}$ is transformed into an enriched version of the same size, as explained later.

Finally, in the classification head, the output of the transformer is first flattened along the sensor dimension. A global average pooling [235] layer is then applied along the temporal dimension. The resulting vector is processed by a fully connected layer with ReLU activation and a dropout layer. The final output probability $p(y_i = 1 \mid \mathsf{X}_i)$ is determined by a fully connected layer with a sigmoid activation function.

[1] The term "head" seems to be originated from the Turing machine, where a head means a tape head to read magnetic tapes.

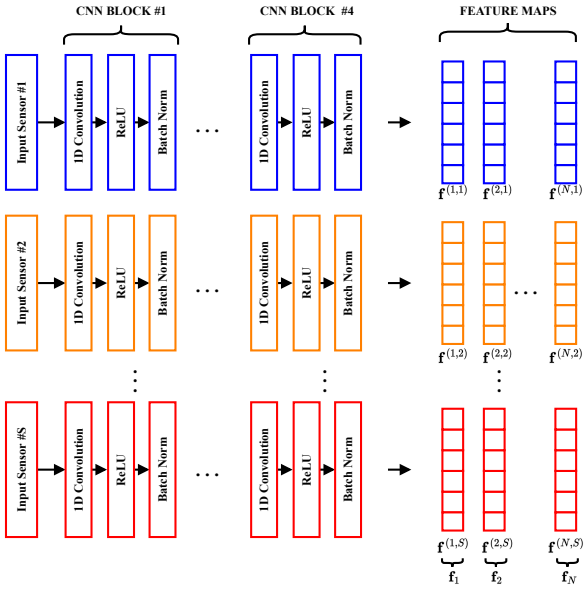


Fig. 6.3: Muti-head 1D CNN.

6.3.2 DFT Encoding

As discussed in Section 6.4 in detail, one issue with the original positional encoding (6.4) is that it has a strong low-pass filtering property. As a result, it puts too much emphasis on long-range correlation among items and tends to suppress short- and mid-range location differences. To address the issue, we propose to use what we call the **faithful-Encoding**:

$$\boldsymbol{e}^{(t)} = \sqrt{\frac{2}{d}} \left(\frac{1}{\sqrt{2}}, \cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_K t), \sin(\omega_K t), \frac{\cos \pi t}{\sqrt{2}} \right)^\top, \tag{6.13}$$

where $\omega_k \triangleq \frac{2\pi k}{d}$ and $K \triangleq \frac{d}{2} - 1$. The derivation is given in Section 6.4. Although the faithful-Encoding seemingly looks similar to the original one (6.4), there is a fundamental difference. One can show that the particular form (6.13) has a mathematical guarantee that it maintains the whole information of the location without introducing any bias.

6.3.3 Learning spatio-temporal dependencies

Now that temporal ordering has been reflected by the faithful-Encoding, we next consider how to reflect ST dependencies in the representation vectors.

Intuitively, this can be done by making highly dependent items have similar representation vectors. Mathematically, it is implemented by defining the transition probability between items in an input sequence.

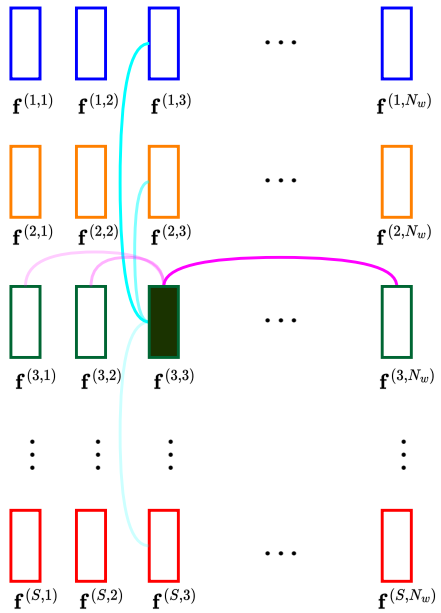


Fig. 6.4: Each rectangular cell within a row (spatial dimension) and a column (temporal dimension) corresponds to an embedding. For a given sensor s and time-segment t (black rectangle), the pink lines denote temporal dependencies, while the blue lines represent spatial dependencies. Notably, the intensity of the color in each cell reflects the strength of the corresponding dependency.

Our basic assumption is that there is a latent dependency structure behind $\{\boldsymbol{f}^{(t,s)}\}$, which is described by the transition probability between two spatio-temporal points (t,s) and (t',s') as $p(t,s \mid t',s')$, as illustrated in Figure 6.4. Here we employ a factorized transition model

$$p(t,s \mid t',s') \approx p(t \mid t',s)p(s \mid s',t), \tag{6.14}$$

where $p(\cdot)$ is used as a symbolic notation representing probability distribution in general rather than a specific functional form. Each component is assumed to be log-quadratic:

$$\ln p(t \mid t',s) = \text{c.} + \frac{1}{\sqrt{M}}(\hat{\boldsymbol{f}}^{(t,s)})^\top \hat{\mathsf{H}}\ \hat{\boldsymbol{f}}^{(t',s)}, \tag{6.15}$$

$$\ln p(s \mid s',t) = \text{c.} + \frac{1}{\sqrt{M}}(\overline{\boldsymbol{f}}^{(t,s)})^\top \overline{\mathsf{H}}\ \overline{\boldsymbol{f}}^{(t,s')}, \tag{6.16}$$

where c. symbolically denotes a constant to meet the normalization condition of the probability distributions. Also, $\hat{\mathsf{H}}, \overline{\mathsf{H}} \in \mathbb{R}^{M \times M}$ are fully learnable matrices that roughly correspond to the precision matrix of a Gaussian diffusion process in the feature space, and $\hat{\boldsymbol{f}}^{(t,s)}, \overline{\boldsymbol{f}}^{(t,s)}$ are the temporal and spatial branches of the embeddings $\boldsymbol{f}^{(t,s)}$, respectively. The quadratic form can also be viewed as a generalized similarity between the pair of feature vectors. In fact, if $\hat{\mathsf{H}}, \overline{\mathsf{H}}$ are identity matrices, the r.h.s. is reduced to the well-known cosine similarity. The logarithm function guarantees the positivity of $p(\cdot)$. For more details on this, see Appendix.

So far, we have assumed that the representation vectors are given. Here, let us consider the opposite: If the transition probability is given, how can we find the representation vectors consistent with the transition probability? In the same spirit of graph neural networks, we can do this by making two-step Markovian transitions:

$$\hat{\boldsymbol{f}}^{(t,s)} \leftarrow \sum_{t'} p(t \mid t', s) \hat{\mathsf{W}}_V \hat{\boldsymbol{f}}^{(t',s)} \qquad \text{for all } (t,s), \tag{6.17}$$

$$\overline{\boldsymbol{f}}^{(t,s)} \leftarrow \sum_{s'} p(s \mid s', t) \overline{\mathsf{W}}_V \overline{\boldsymbol{f}}^{(t,s')} \qquad \text{for all } (t,s), \tag{6.18}$$

where $\hat{\mathsf{W}}_V, \overline{\mathsf{W}}_V \in \mathbb{R}^{M \times M}$ is a fully learnable parameter matrix to absorb indistinguishably due to scaling and rotation. As shown in Figure 6.2, both $\hat{\boldsymbol{f}}^{(t,s)}$ and $\overline{\boldsymbol{f}}^{(t,s)}$ are added, obtaining an enriched representation $\boldsymbol{f}^{(t,s)}$. By this diffusion process, the higher the transition probability is between a pair of the representation vectors, the more similar they are.

*A. **Relationship with Transformer***

Our ST dependency model can be viewed as a variant of the Transformer [28]. To see this, consider the singular value decompositions (SVD) of the temporal and spatial branches $\hat{\mathsf{H}} = \hat{\mathsf{W}}_Q \hat{\mathsf{W}}_K^\top$ and $\overline{\mathsf{H}} = \overline{\mathsf{W}}_Q \overline{\mathsf{W}}_K^\top$, where $\hat{\mathsf{W}}_Q$, $\hat{\mathsf{W}}_K$, $\overline{\mathsf{W}}_Q$ and $\overline{\mathsf{W}}_K$ are the left and right singular matrices with column vectors normalized to the square root of the singular values. Since any well-defined matrices have an SVD, without loss of generality, Eqs. (6.17)-(6.18) are reduced to the well-known query-key-value mapping of the Transformer:

$$\hat{\mathsf{F}}^{(s)} \leftarrow \text{softmax}\left(\frac{1}{\sqrt{M}} \hat{\mathsf{Q}} \hat{\mathsf{K}}^\top\right) \hat{\mathsf{V}} \tag{6.19}$$

$$\overline{\mathsf{F}}^{(t)} \leftarrow \text{softmax}\left(\frac{1}{\sqrt{M}} \overline{\mathsf{Q}}\overline{\mathsf{K}}^\top\right) \overline{\mathsf{V}} \tag{6.20}$$

where $\mathsf{F}^{(s)} \triangleq [\boldsymbol{f}^{(1,s)}, \ldots, \boldsymbol{f}^{(N_w,s)}]$, $\hat{\mathsf{Q}} \triangleq \hat{\mathsf{W}}_Q \mathsf{F}^{(s)}, \hat{\mathsf{K}} \triangleq \hat{\mathsf{W}}_K \mathsf{F}^{(s)}, \hat{\mathsf{V}} \triangleq \hat{\mathsf{W}}_V \mathsf{F}^{(s)}$ and $\mathsf{F}^{(t)} \triangleq [\boldsymbol{f}^{(t,1)}, \ldots, \boldsymbol{f}^{(t,S)}]$, $\overline{\mathsf{Q}} \triangleq \overline{\mathsf{W}}_Q \mathsf{F}^{(t)}$, $\overline{\mathsf{K}} \triangleq \overline{\mathsf{W}}_K \mathsf{F}^{(t)}, \overline{\mathsf{V}} \triangleq \overline{\mathsf{W}}_V \mathsf{F}^{(t)}$. After this, the enriched representations are added, normalized, and represented in a single representation vector:

$$\mathsf{F} \leftarrow \text{LayerNorm}(\mathsf{F} + \text{Dropout}(\hat{\mathsf{F}})) + \text{LayerNorm}(\mathsf{F} + \text{Dropout}(\overline{\mathsf{F}})) \quad (6.21)$$

After the aggregation, the enriched representation F is normalized after being passed to a 2-layer feedforward network with ReLU activation function and a dropout layer:

$$\mathsf{F} \leftarrow \text{LayerNorm}(\mathsf{F} + \text{Dropout}(\text{ReLU}\left((\mathsf{F}\mathsf{W}_1 + b_1)\,\mathsf{W}_2 + b_2\right))) \quad (6.22)$$

where $\mathsf{W}_1, \mathsf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ and $b_1, b_2 \in \mathbb{R}^{d_{ff}}$ are the weights and biases of the first and second feedforward layers, respectively. The dimension of the feedforward layer is set to $d_{ff} = 2048$. A single transformer layer is used in the experiments.

6.3.4 Evaluation of attention.

As the model has been trained for anomaly detection, a high attention value should indicate that the model has given more importance to a given embedding in deciding whether it is anomalous. To evaluate if high attention means high impact in the model decision, we propose the *Attention Trustworthiness Score* (*AT-Score*). The AT-Score evaluates the effectiveness of the attention mechanism in focusing correctly on when and where the anomalies have occurred by measuring how much the model prediction changes when the top-k percent of the spatial and temporal attention weights are set to zero, and it is defined as

$$\text{AT}_k = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \{\hat{y}(\mathsf{X}_i) - \hat{y}^k(\mathsf{X}_i)\} \quad (6.23)$$

where $\hat{y}(\mathsf{X}_i)$ denotes the output of the model before sigmoid and $\hat{y}^k(\mathsf{X}_i)$ denotes the output of the model before sigmoid and setting the top-k percent of attention weights to zero.

6.3.5 Diagnostic scores

The learned transition probabilities allow a variety of diagnosis tasks. Let A and B be the temporal and spatial attention matrices. Based on these matrices we define some scores for temporal and spatial diagnosis.

For temporal diagnosis, we define four scores:

$$\mathsf{A}^{(s)}_{t',t} \triangleq p(t \mid t', s), \quad a^{(s)}_t \triangleq \sum_{t'=1}^{N_w} \mathsf{A}^{(s)}_{t',t}, \quad (6.24)$$

$$\mathsf{A}^{\text{G}}_{t',t} \triangleq \frac{1}{S} \sum_{s=1}^{S} \mathsf{A}^{(s)}_{t',t}, \quad a^{\text{G}}_t \triangleq \frac{1}{S} \sum_{s=1}^{S} a^{(s)}_t, \quad (6.25)$$

where $a_t^{(s)}$ is the influence of a time point t measured locally at the s-th sensor, $\mathsf{A}^G_{t',t}$ is the global influence of t' on t, and a_t^{G} is the global influence of t across all the variables.

For spatial (i.e., variable-variable) diagnosis, we define:

$$\mathsf{B}^{(t)}_{s',s} \triangleq p(s \mid s', t), \quad b_s^{(t)} \triangleq \sum_{s'=1}^{S} \mathsf{B}^{(t)}_{s',s}, \tag{6.26}$$

$$\mathsf{B}^{\mathrm{G}}_{s',s} \triangleq \frac{1}{N_w} \sum_{t=1}^{N_w} \mathsf{B}^{(t)}_{s',s} a_t^{\mathrm{G}}, \quad b_s^{\mathrm{G}} \triangleq \sum_{s'=1}^{S} \mathsf{B}^{G}_{s',s}, \tag{6.27}$$

where $b_s^{(t)}$ is the influence of the variable s at a time t, $\mathsf{B}^G_{s',s}$ is the global influence of the variable s' on s, and b_s^{G} is the global influence of the variable s.

6.4 Fourier Analysis of Positional Encoding

In this section, we discuss how Fourier analysis is used to evaluate the goodness of positional encoding (PE) and derive the faithful-Encoding (6.13).

6.4.1 Original PE has a strong low-pass property

Positional encoding in the present setting is the task of finding a d-dimensional representation vector of an item in the input sequence of length N_w. In the vector view, the position of the s-th item in the input sequence is most straightforwardly represented by an N_w-dimensional "one-hot" vector, whose s-th entry is 1 and otherwise 0. Unfortunately, this is not an appropriate representation because the dimensionality is fixed to be N_w, and it does not have continuity over the elements at all, which makes numerical optimization challenging in stochastic gradient descent.

The original PE in Eq. (6.4) is designed to eliminate these limitations. Then, the question is how we can tell the goodness of its specific functional form. One approach suggested by the sinusoidal form is to use DFT. Consider DFT on the 1-dimensional (1D) lattice with d lattice points ($d > N_w$). Any function defined on this lattice is represented as a linear combination of the sinusoidal function with the frequencies $\{\omega_k\}$. It is interesting to see how the frequencies $\{w_k\}$ in Eq. (6.4) are distributed in the Fourier space.

The black line in Fig. 6.5 shows the distribution of the frequencies of Eq. (6.4). The distribution is estimated with the kernel density estimation with the Gaussian kernel [236]. Specifically, at each ω_k, the weight is given by

$$g_k = \sum_{l \in \{0,2,\ldots,d-2\}} \frac{1}{R} \exp\left(-\frac{1}{2\sigma^2}(\omega_k - w_l)^2\right), \tag{6.28}$$

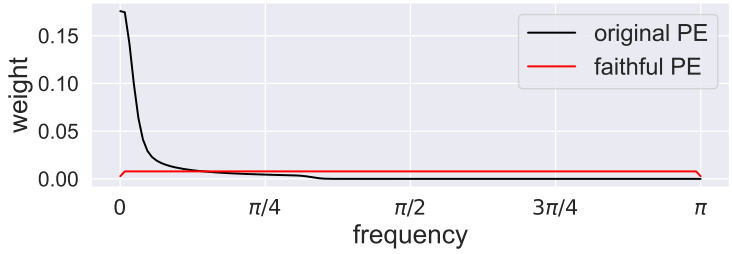


Fig. 6.5: The distribution of the frequency $w_k = \rho^{-\frac{k}{d}}$ in Eq. (6.4) over the Fourier bases (black line), where $\rho = 10\,000, d = 256$. Notice the contrast to that of the faithful-Encoding, which gives a uniform distribution (red line; See Section 6.4.3).

where R is a normalization constant to satisfy $\sum_k g_k = 1$. We chose the bandwidth $\sigma = 4 \times \frac{2\pi}{d}$ with $d = 256$. Due to the power function $\rho^{-\frac{k}{d}}$, the distribution is extremely skewed towards zero.

This fact can be easily understood also by running a simple analysis as follows. The first and second smallest frequencies are $0, \frac{2\pi}{d}$, respectively. We can count the number of w_ks that fall into between them. Solving the equation

$$\frac{2\pi}{d} = \rho^{-\frac{l}{d}} \tag{6.29}$$

and assuming $\rho = 10\,000$, we have $l = \frac{d}{4} \log_{10} \frac{d}{2\pi} \approx 103$ for $d = 256$ and $l \approx 245$ for $d = 512$. Hence, almost a half of the entries go to this lowest bin.

This simple analysis, along with Fig. 6.5, demonstrates that the original PE has a strong bias to suppress mid and high frequencies. As a result, it tends to ignore mid- and short-range differences in location. This can be problematic in applications where short-range dependencies matter, such as time-series classification for physical sensor data.

6.4.2 Original PE lacks faithfulness

Another interesting question is what kind of function the skewed distribution g_k represents. To answer this question, we perform an experiment described in Algorithm 1, which is designed to understand what kind of distortion it may introduce to an assumed reference function. In our PE context, the reference function should be the position function (a.k.a. one-hot vector) since the original PE (6.4) was proposed to be a representation of the item at the location s.

Figure 6.6 shows the result of reconstruction. We normalized the modified DFT coefficients so the original ℓ_2 norm is kept unchanged. All the parameters used are the same as those in Fig. 6.5, i.e., $d = 256, N_w = 80, h = 10000, \sigma =$

Algorithm 1 Reference function reconstruction

Require: Reference function $f(t)$, DFT component weights $\{g_k\}$.
1: Find DFT coefficients of f as $\{(a_k, b_k) \mid k = 0, \ldots, K\}$.
2: $a_0 \leftarrow a_0 g_0$ and $b_0 \leftarrow b_0 g_{K+1}$
3: **for all** $k = 1, \ldots, K$ **do**
4: $\quad a_k \leftarrow a_k g_k$ and $b_k \leftarrow b_k g_k$
5: **end for**
6: Inverse-DFT from the modified coefficients.

$4 \times \frac{2\pi}{d}$. As expected from the low-pass property, the reconstruction by the original PE failed to reproduce the delta functions. The broad distributions imply that the original PE is not sensitive to the difference in the location up to about 30. As the total sequence length is $N_w = 80$, we conclude that the original PE tends to put an extremely strong emphasis on global long-range dependencies within the sequence.

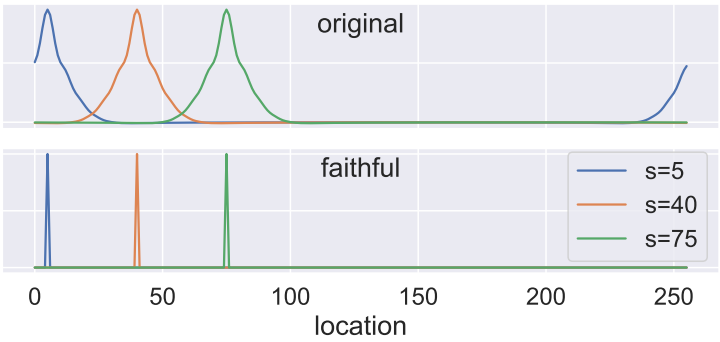


Fig. 6.6: Reconstruction by Algorithm 1 for the location function $f(t) = \delta_{t,5}, \ \delta_{t,40}, \ \delta_{t,75}$. Perfect reconstruction corresponds to single-peaked spikes at $t = 5, 40, 75$, respectively. The broad distributions in the top panel demonstrate a significant loss of information in the original PE. See Section 6.4.3 for the faithful-Encoding (bottom).

If the PE is supposed to get a representation of the position function, PE should faithfully reproduce the original position function. Here, we formally define the notion of the faithfulness of PE:

Definition 27 (Faithfulness of PE). *A positional encoding is said faithful if it is injective (i.e., one-to-one) to the position function.*

For PE, the requirement of faithfulness seems natural. Very interestingly, as long as the sinusoidal bases are assumed, this requirement almost auto-

matically leads to a specific PE algorithm that we call the faithful-Encoding, which is the topic of the next subsection.

6.4.3 DFT-based derivation of positional encoding

For the requirement of faithfulness, one straightforward approach is to leverage DFT for positional encoding. The idea is to use the DFT representation as a smooth surrogate for the one-hot function $f_s(t) \triangleq \delta_{s,t}$, where $\delta_{s,t}$ is Kronecker's delta giving 1 only if $s = t$ and 0 otherwise. Let

$$a_0^{(s)}, a_1^{(s)}, \ldots, b_1^{(s)}, \ldots, b_K^{(s)}, b_0^{(s)}$$

be the DFT coefficients of $f_s(t)$. It is straightforward to compute the coefficients against the real Fourier bases $\{\varphi_k(t)\}$:

$$a_0^{(s)} = \sum_{t=0}^{d-1} \delta_{s,t}\varphi_0(t) = \sum_{t=0}^{d-1} \delta_{s,t}\frac{1}{\sqrt{d}} = \frac{1}{\sqrt{d}}, \tag{6.30}$$

$$a_1^{(s)} = \sum_{t=0}^{d-1} \delta_{s,t}\varphi_1(t) = \sum_{t=0}^{d-1} \delta_{s,t}\sqrt{\frac{2}{d}}\cos(\omega_1 t) = \sqrt{\frac{2}{d}}\cos(\omega_1 s), \tag{6.31}$$

$$\vdots$$

$$b_K^{(s)} = \sum_{t=0}^{d-1} \delta_{s,t}\varphi_K(t) = \sum_{t=0}^{d-1} \delta_{s,t}\sqrt{\frac{2}{d}}\sin(\omega_K t) = \sqrt{\frac{2}{d}}\sin(\omega_K s), \tag{6.32}$$

$$b_0^{(s)} = \sum_{t=0}^{d-1} \delta_{s,t}\varphi_{d-1}(t) = \sum_{t=0}^{d-1} \delta_{s,t}\sqrt{\frac{1}{d}}\cos(\pi t) = \sqrt{\frac{1}{d}}\cos(\pi s), \tag{6.33}$$

where $\varphi_k(t)$s are defined by the second equality. Simply using the DFT coefficients, we define the vector representation of the one-hot function as

$$\boldsymbol{e}^{(s)} \triangleq (a_0^{(s)}, a_1^{(s)}, \ldots, b_1^{(s)}, \ldots, b_K^{(s)}, b_0^{(s)})^\top, \tag{6.34}$$

which is exactly the same as the faithful-Encoding (6.13).

Because of the general properties of DFT, the following claim is almost evident:

Theorem 1. *The DFT-based encoding* (6.34) *is faithful.*

Proof. This follows from the existence of inverse transformation in DFT. We can also prove it directly. With Eq. (6.34), the r.h.s. of the DFT expansion

$$f_s(t) = \frac{1}{\sqrt{d}}[a_0 + b_0\cos(\pi t)] + \sqrt{\frac{2}{d}}\sum_{k=1}^{K}\left(a_k\cos(\omega_k t) + b_k\sin(\omega_k t)\right), \tag{6.35}$$

is given by

$$\text{r.h.s.} = \frac{1}{d}[1 + (-1)^{s-t}] + \frac{2}{d}\sum_{k=1}^{K}\cos(\omega_k(s-t))$$

It is obvious that the r.h.s. is 1 if $s = t$. Now, assume $s \neq t$. Using $\cos x = \frac{1}{2}(\mathrm{e}^{\mathrm{i}x} + \mathrm{e}^{-\mathrm{i}x})$, where i is the imaginary unit, and the sum rule of geometric series, we have

$$\text{r.h.s.} = \frac{1}{d}[1 + (-1)^{s-t}] + \frac{1}{d}\frac{c_{s,t} - c_{s,t}^{K+1} - 1 + c_{s,t}^{-K}}{1 - c_{s,t}},$$

where we have defined $c_{s,t} \triangleq \exp\left(\mathrm{i}\frac{2\pi(s-t)}{d}\right)$. By noting $c_{s,t}^{-2K-1} = c_{s,t}$ and $c_{s,t}^{K+1} = (-1)^{s-t}$, it is straightforward to show the r.h.s. is 0. Putting all together, the inverse DFT of the faithful-Encoding gives $\delta_{s,t}$, which is the location function. □

In Fig. 6.5, we have shown the distribution of the faithful-Encoding in the Fourier domain. From Eqs. (6.30)-(6.33), we see the distribution is given by

$$g_k^{\mathrm{DFT}} = \frac{1}{d}(\delta_{k,0} + \delta_{k,d-1}) + \frac{2}{d}\sum_{l=1}^{K}\delta_{k,l}, \tag{6.36}$$

where $\delta_{k,0}$ etc. are Kronecker's delta function. This distribution is flat except for the terminal points at $k = 0, d-1$. Hence, unlike the original PE, the proposed faithful PE does not have any bias on the choice of the Fourier components.

With this flat distribution, we also did the reconstruction experiment. The result is shown in the bottom panel in Fig. 6.6. We see that the location functions are perfectly reconstructed with the faithful PE.

6.5 Experiments

In this section, we want to demonstrate the ability of DFStrans to detect and diagnose anomalies. To do so, we compare the method with four other baseline algorithms on four datasets.

6.5.1 Datasets and baselines

A. ***Industrial Case Study***

The industrial dataset used in this study was obtained from a physical model that simulates the behavior of an elevator during up and down journeys. The physical model was designed by a domain expert from a collaborating company

that specializes in the manufacturing of elevators. This model was designed to produce fault conditions during these journeys, such as reduced lubrication, misalignment, or localized peaks in the guidance system, or demagnetization or loss of inductance in the electric machine. In this study, three effects related to misalignment, reduced lubrication, and localized peaks in the guidance system were introduced into the model. The first two effects increase the friction between the cabin or counterweight and the rails, and are referred to as *friction journeys (FJs)*. Localized bumps cause local impacts on the cabin, resulting in spike-like deformations, which are referred to as *point anomalous journeys (PAJs)*. Table 6.1 summarizes the sensors installed in the elevator.

Table 6.1: Summary of the sensors installed in the elevator.

Sensor name	**Description**
Alpha	Angular acceleration of the pulley
Ax, Ay, Az	Lateral acceleration of cabin on X, Y, and Z axis
Fc, Fcw	Tension on the cabin's and counterweight cable
FrictionCabin, FrictionCw	Cabin and counterweight friction
Fsupport	Force on the support of the machine-pulley
Id, Iq	Direct and quadrature power
Omega	Angular speed of the pulley
Phi	Angular position of the pulley
PulleyAz	Vertical acceleration of the pulley
Vc, Vcw	Cabin and Counterweight speed
Vd, Vq	Direct and quadrature voltage
Zc, Zcw	Cabin and counterweight position

*B. **Baseline algorithms***

The goal of this work is mainly based on proposing a solution for anomaly diagnosis, but without losing classification performance. Therefore, we compare with some of these state-of-the-art time series classification models. However, these algorithms are not meant for anomaly diagnosis. We compare our classification performance against: (1) **MH1DCNN-LSTM** [234], (2) **TapNet** [237], (3) **InceptionTime** [238] and (4) **MLSTM-FCN** [239]. See Appendix for details on the model's hyperparameters.

*C. **Benchmark datasets***

We evaluated the methods in three other datasets apart from the elevator use-case: both (1) **Soil Moisture Active Passive satellite (SMAP)** [240] and (2) **Mars Science Laboratory rover (MSL)** [240] are public datasets

Table 6.2: Results obtained with the models for the different datasets in terms of Precision (**P**), Recall (**R**) and F1-Score (**F1**). Best scores are highlighted in **bold**.

Datasets	Methods														
	MH1DCNN-LSTM			**InceptionTime**			**TapNet**			**MLSTM-FCN**			**DFStrans**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Elevator	0.993	0.914	0.951	**1**	0.892	0.944	0.398	0.711	0.510	0.913	0.724	0.808	0.989	**0.917**	**0.952**
SMD	0.913	0.744	0.807	0.856	0.925	0.882	0.842	0.606	0.706	0.951	0.853	0.895	**0.973**	**0.946**	**0.959**
MSL	0.821	0.821	0.821	**0.965**	**0.948**	**0.956**	0.875	0.867	0.857	0.952	0.925	0.938	0.867	0.882	0.874
SMAP	0.923	0.805	0.859	0.853	0.687	0.761	0.66	**0.908**	0.765	**0.963**	0.833	**0.894**	0.883	0.826	0.853
mean	0.912	0.821	0.859	0.918	0.863	0.888	0.694	0.773	0.709	**0.945**	0.834	0.884	0.931	**0.893**	**0.911**

provided by NASA that consist of telemetry data from spacecraft monitoring systems, and (3) **Server Machine Dataset (SMD)** [185] is a dataset collected from a large Internet company. These three datasets are unsupervised, so the labeled testing data have been used to validate the models.

D. ***Data preprocessing and evaluation***

Regarding data preprocessing, all data have been scaled (independently for each sensor) between 0 and 1, using MinMaxScaler module of Sklearn [241]. For benchmark datasets, we have divided the data into sub-series X_i of $T = 500$ points and labeled them as 1 if an anomaly has occurred during that period and 0 otherwise. For algorithms using multi-head CNNs, these time-series have been divided into non-overlapping windows of length w_l, i.e. $\mathsf{X}_i = \{\mathsf{X}_i^{(1)}, ..., \mathsf{X}_i^{(N_w)}\}$. We have used a 5-fold cross-validation strategy for training and evaluation, using 70% for training, 15% for validation, and 15% for testing.

6.5.2 Anomaly detection

Our model was evaluated on four datasets using four baseline algorithms. As shown in Table 6.2, for the elevator use case, DFStrans was the best performer in terms of Recall and F1. Additionally, DFStrans was the top performer on the SMD dataset in all terms. On the MSL dataset, InceptionTime achieved the best results, outperforming the other models. Finally, on the SMAP dataset, MLSTM-FCN performed the best in terms of F1, followed by DFSTrans and MH1DCNN-LSTM. Overall, our algorithm achieved the best average results in terms of Recall and F1, indicating that DFStrans is a competitive algorithm for anomaly detection. It is worth noting that our algorithm also focuses on diagnosing anomalies rather than providing competitive results for anomaly detection.

A. ***Ablation on positional encoding***

Here we compare the results obtained using Vaswani's positional encoder and using the faithful-Encoding. As shown in Table 6.3, the faithful-Encoding

achieves better results than Vaswani's positional encoding in three of the four datasets, demonstrating the importance of faithful-Encoding.

Table 6.3: Results obtained with the models for the different datasets in terms of Precision (**P**), Recall (**R**), and F1-Score (**F1**) under different encoding strategies. Best scores are highlighted in **bold**.

Datasets	Encoding strategies					
	w Vaswani's encoding			w faithful-Encoding		
	P	R	F1	P	R	F1
Elevator	0.952	0.914	**0.931**	**0.989**	**0.917**	**0.952**
SMD	**0.986**	0.942	**0.963**	0.985	**0.948**	**0.965**
MSL	0.862	0.838	0.849	**0.867**	**0.882**	**0.874**
SMAP	**0.887**	0.823	**0.854**	0.883	**0.826**	0.853
mean	0.922	0.879	0.899	**0.931**	**0.893**	**0.911**

6.5.3 Anomaly diagnosis

Next, we evaluate the capability in anomaly diagnosis using the elevator's use case, in which expert feedback is available. Here, we show a few examples of how the temporal and spatial attention matrices, A and B respectively (defined in Section A.), are used to obtain intrinsic interpretability for diagnosing the detected anomalies.

In Figure 6.7, we present the spatial attention matrix and spatial relevance vector for a PAJ containing two point anomalies, as identified by a domain expert, at time segments $t = 34$ and $t = 58$, reflected in the accelerations Ax and Ay. Only a subset of sensors is shown for space limitation. The local spatial interpretations allow us to investigate the sensor that the model is attending to at each time segment. We selected the time segments where the anomalies occur, $t = 34$ and $t = 58$, and plotted the spatial attention matrix $\mathsf{B}^{(t)}$ with the relevance scores $b_s^{(t)}$ of each sensor s to examine whether the model is focusing on the accelerations. We also randomly selected another time segment, $t = 75$, where there is no anomaly, to examine the behavior of the spatial attention at that point. In time segments $t = 34$ and $t = 58$ (Figures 6.7c and 6.7d), we observe that the relevance of the accelerations Ax and Ay is high while the relevance of the other sensors is low, as expected. In contrast, for time segment $t = 75$, all sensors have similar relevance, and the model does not focus on any particular sensor. We also observe that the global spatial relevances are high for both accelerations.

Unlike in PAJs, in FJs, friction is usually present during the whole journey. Therefore, the attention should be somewhat distributed among all-time segments, with potentially stronger attention during the acceleration and deceleration phases according to the expert's observations. This is demonstrated

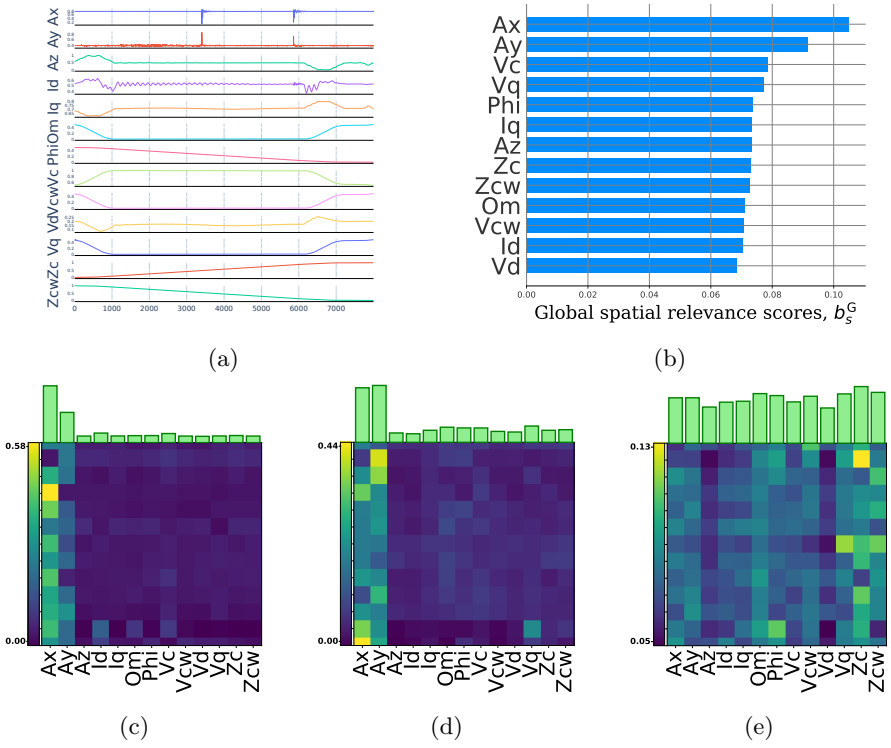


Fig. 6.7: The first plot **(a)** shows a PAJ containing two point anomalies, at $t = 34$ and $t = 58$, which can be seen as ripples in the accelerations. Then, the plot **(b)** shows the global spatial relevances. Plots **(c)**, **(d)** and **(e)** show the spatial attention matrix and the spatial relevance vectors for time steps $t = 34$, $t = 58$ and $t = 75$, respectively.

in Figure 6.8, where the global temporal attention matrix and global temporal attention relevances show that attention is present throughout the entire journey, with slight peaks at the beginning (acceleration) and end (deceleration) of the journey, rather than any individual time segment standing out significantly as in PAJs. Additionally, FJs tend to increase the required motor torque and electrical consumption, as reflected in the sensors measuring electrical current (i.e., Iq and Id) and the electric machine voltages (i.e., Vq and Vd). The global spatial attention matrix and global spatial relevance scores in Figure 6.8 show that these sensors had the greatest influence on the model's classification of the journey as anomalous.

Figure 6.9 presents the temporal attention matrix and spatial relevance vectors. The first two plots depict the local temporal relevances for five different PAJs. Accelerations are plotted because, according to the domain expert,

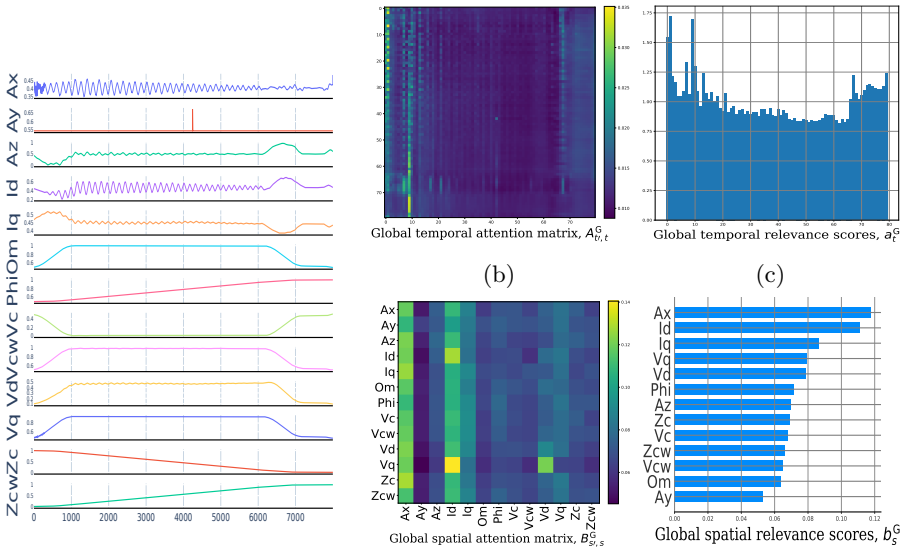


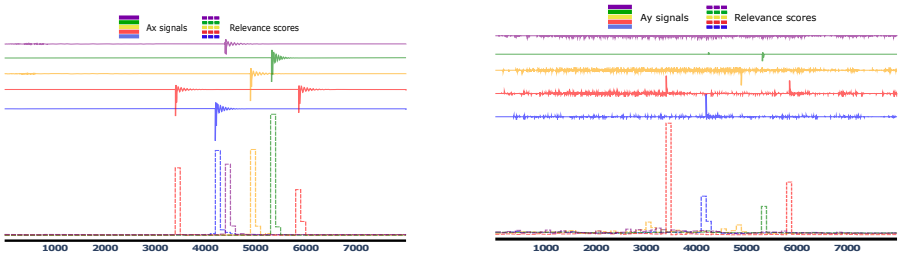
Fig. 6.8: **(a)** shows a FJ. Then, **(b)** shows the global temporal attention matrix, **(c)** the global temporal attention relevances, **(d)** the global spatial attention matrix and **(e)** the global spatial attention relevances.

anomalies in PAJs are typically reflected in these sensors. Figure 6.9c shows the temporal attention of the Iq sensor on six different journeys, three of which are normal (shown in green) and three of which are FJs with the *Iq* sensor in the top positions (shown in red). In these cases, the distinction between anomalous and non-anomalous journeys is most apparent during the acceleration and deceleration phases, where the attention is stronger in the FJs. Additionally, the temporal relevance is evenly distributed throughout the entire journey.

*A. **Evaluating Attention***

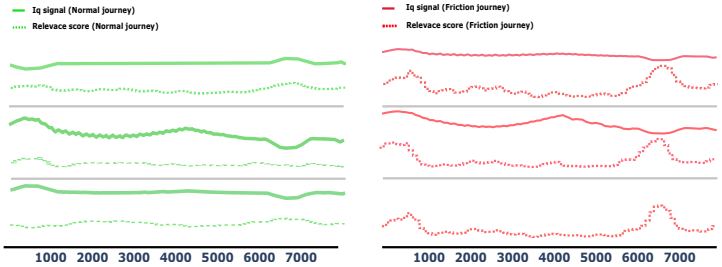
To quantify the trustworthiness of the attention mechanisms, we have used the *AT-Score* metric under different percentages of top-k attention weights set to zero. Since the PAJs exhibit anomalous behavior at specific time steps during the journey and the FJs exhibit anomalous behavior throughout the entire time series, we have distinguished both to conduct this experiment.

Figure 6.10a shows that as the weights with the highest attention values in the PAJs are set to zero, the value of the model output decreases. This trend can be seen in the medians and means of these values, represented as red and green lines. This means that the PAJs are classified as normal journeys if the attention mechanism does not focus on the most relevant time segments,



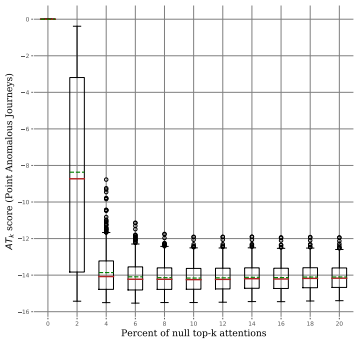
(a)

(b)

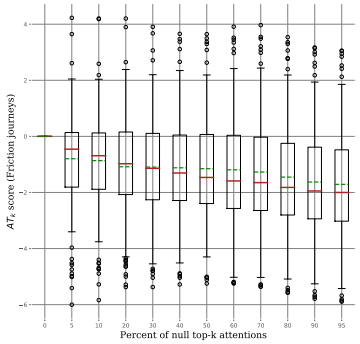


(c)

Fig. 6.9: Local temporal attention plots. Figures **(a)** and **(b)** are plots of five PAJs, in which the observations of the Ax and Ay sensors are shown together with their temporal relevance scores $a^{(s)}$. Figure **(c)** is a comparison of the temporal relevance scores for sensor Iq between three FJs (red lines) and three normal journeys (green lines).



(a)



(b)

Fig. 6.10: Boxplots of the AT-Scores obtained under different percentages of top-k attention weights set to zero. Plot **(a)** corresponds to PAJs and plot **(b)** to corresponds FJs. Median and mean values are represented with red and green lines, respectively.

which means that it was previously focusing on where the anomalous behavior occurred during the journey. However, as shown in Figure 6.10b, this effect is less accentuated and more variable in FJs, because this type of anomalous behavior can be noticed throughout the whole journey.

6.6 DFTrans, a reliable supervised solution for anomaly detection and diagnosis

With this research, we have written an article that has been accepted in the journal Knowledge-Based Systems:

Article 3 (A3): Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Diagnostic Spatio Temporal Transfomer with Faithfull Encoding*. Knowledge-Based Systems. **Status**: *Accepted*.

This research presents a new framework that leverages a spatio-temporal dependency model for diagnosing anomalies in noisy multivariate sensor data. The framework combines multi-head 1D CNNs and a Transformer-like spatio-temporal architecture. The multi-head 1D CNNs are used to obtain rich embeddings from raw sensor data while preserving the temporal nature of the data. Then, the Transformer-like architecture allows learning the spatial and temporal dependencies between these embeddings. Additionally, the study mathematically demonstrates the limitations of the positional encoding used in vanilla Transformers, which disproportionately emphasizes long-range correlations and suppresses short- and medium-term location differences, which can be detrimental in anomaly detection scenarios where small changes over time are important. To address this issue, we propose a DFT-based positional encoding, called faithful encoding, that does not introduce any bias in the information about the location of the items and has a theoretical guarantee of faithfulness.

The effectiveness of the proposed framework was demonstrated through evaluations on four different datasets and an ablation study comparing the results using vanilla and faithful positional encodings. The results showed that **the faithful encoding improved upon the vanilla encoding in most benchmark datasets, and a sensitivity analysis demonstrated the reliability of attention in diagnosing anomalies.** Overall, the proposed **DFSTrans framework offers reliable solutions for detecting and diagnosing anomalies in multi-sensor industrial scenarios.**

In this research, we have made substantial strides toward our objective of creating intrinsically interpretable models tailored for anomaly detection. Our contributions span multiple areas, including the **introduction of faithful encoding, the development of a robust model architecture that effectively integrates CNNs and Transformers, and the introduc-**

tion of new metrics based on attention mechanisms for diagnosing anomalies.

However, these contributions have been examined within a supervised learning framework. While this approach provides a structured and controlled setting to test the capabilities of our model, **it inherently suffers from a significant limitation. The acquisition of labeled anomalies, which is a prerequisite for supervised learning, often poses a real-world challenge, due to the rarity of anomalous events.**

Recognizing this challenge, we seek to extend our research into the domain of unsupervised learning, where we are not confined by the need for labeled data. In our second contribution, **we aim to adapt the developed DFSTrans framework for an unsupervised setting, thereby making it more practical and applicable for a wider array of real-world anomaly detection tasks.**

7

Unsupervised Anomaly Diagnosis with Masked Spatio-Temporal Transformers

In this section, we describe our second contribution related to the **use of transformer models for anomaly detection**. Considering the positive results obtained in the supervised setting, we introduce an unsupervised version of DFSTrans, named **Unsupervised Diagnostic Spatio Temporal Transformer** (uDFSTrans).

Although our supervised version, DFSTrans, reported promising results in diagnosing anomalies, it is not always possible to use supervised approaches for anomaly detection due to the difficulty of having labeled anomalies. Therefore, it is necessary to have an unsupervised approach that effectively handles these spatio-temporal dependencies inherent in the data. Several recent studies have proposed different strategies to tackle anomaly detection tasks using transformers in an unsupervised way [242, 243, 212]. These methodologies address the anomaly detection problem as one of prediction. Given a sequence of values, these approaches predict future values. Anomalies are then identified using an anomaly score, which is typically based on the error in reconstruction, and compared against a predefined threshold.

In prediction tasks utilizing transformers, masking is typically applied to the values being predicted. This masking usually takes place before applying the *softmax* function over the dot-product between query and key values. However, we argue that this type of masking may not be entirely appropriate, as the values to be predicted are implicitly exposed to the model when applying this dot product. We propose that a different masking strategy may be needed to fully obscure the future values to be predicted. Moreover, as demonstrated in previous work, current transformers employ sinusoidal positional encoding, which struggles to capture high frequencies. This suggests that existing transformers might not be fully optimized to effectively tackle anomaly detection tasks.

With uDFSTrans, we approach the anomaly detection task from a different perspective. On the one hand, we argue that in order to consider a series of values within a sequence as anomalous, the context in which these values exist must be taken into account. To this end, we propose a context-based embed-

ding generator, which effectively masks the values to be predicted by creating context embeddings, thereby capturing the information of the surrounding context. We suggest the implementation of a Global Alignment Transformer (GAT) as a part of the proposed architecture, which is detailed in Section 7.3.1.

On the other hand, we employ a multi-masking strategy, using varied masking lengths to forecast upcoming values. We believe that anomalies pose a greater challenge for accurate reconstruction compared to regular values. As the masking length increases, the challenge of precise prediction intensifies. Consequently, we theorize that there will be notable disparities in the reconstruction errors when comparing values predicted with different mask lengths.

Finally, we believe that the attention mechanisms of the transformer should behave differently when there is an anomaly, and that variations between these connections will be greater when anomalies occur. To model this, we propose an anomaly score that takes into account the reconstruction errors, the differences between errors for different maskings, and the variations among the attention mechanisms.

7.1 Problem setting

In this section, we define the problem addressed in this work by uDFSTrans, and explain the main idea behind it.

Unlike the previous supervised approach where we were provided with N sets of S-variate time-series with binary labels, in this case, we are dealing with a single S-variate time-series $\mathcal{X} = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_T\}$ defined as a sequence of ordered, real-valued observations of length T, where at each timestamp t, $\boldsymbol{x}_t \in \mathbb{R}^S$ represents a set of points collected from S different sensors. Similarly, we can define a corresponding label sequence $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$ for the time-series $\mathcal{X}$.

In this instance, we aim to predict these labels in an unsupervised manner, utilizing a common practice often employed in anomaly detection and diagnosis. Firstly, we divide the time-series into smaller windows of length K, i.e.,

$$\boldsymbol{w}_t = \{\boldsymbol{x}_{t-K+1}, \ldots, \boldsymbol{x}_t\}$$

This is done to enable a more comprehensive consideration of the contextual information associated with each data point $\boldsymbol{x}_t \in \mathcal{X}$. In this context, the anomaly detection model $\mathcal{M}$ is trained using the windows instead of $\mathcal{X}$. However, when attempting to predict the label of a given point $\boldsymbol{x}_t$, we consider the model $\mathcal{M}$ not only to take into account the immediate window $\boldsymbol{w}_t$ as input but also a set of N windows, defined as

$$\mathsf{W}_t = \{\boldsymbol{w}_{t-N+1}, \ldots, \boldsymbol{w}_t\}$$

that enable the identification of long-term dependencies. Specifically, our approach involves leveraging multiple windows of past observations in order to provide a more comprehensive understanding of the context surrounding the target datapoint $\boldsymbol{x}_t$.

To predict the labels $\boldsymbol{y}_t$, the initial step is to calculate an anomaly score $\mathbf{s}_t$. Once the anomaly score has been computed, the predicted labels are obtained by binarizing the scores using a threshold value t, i.e.,

$$y_t = \mathbb{1}\left(\mathbf{s}_t \geq t\right).$$

A. How to obtain the scores.

Without delving into the specifics of how to obtain anomaly scores (a detailed description can be found in Section 7.3.2), we introduce the main idea. An anomaly can be a sudden variation in one or several sensors, it may propagate over time, or it may involve certain patterns in one or multiple variables that, depending on the context in which they occur, can be considered anomalous. In this work, we aim to address the problem of anomaly detection using transformers. The objective is for the model to accurately predict the values of the window $\boldsymbol{w}_t$ when it is normal and to perform poorly when it contains anomalies. To achieve this goal, different-sized maskings are utilized, and the anomaly score takes into account factors such as the overall reconstruction error, differences in reconstruction between different maskings, and variations in attention mechanisms.

7.2 Analyzing masking strategy of Transformers

In this section, we analyze the masking strategy typically employed in transformers.

In general, using transformers for prediction requires the use of masking to prevent *cheating* in the decoder. Typically, this masking is accomplished by adding to the product of query and key values a diagonal matrix $N \times N$ matrix M with -∞ values on the diagonal. Adding M, when applying the softmax function the resulting attention matrix has zero values on the diagonal, which prevents the model from attending to these positions during training.

$$\mathsf{A} \triangleq \operatorname{softmax}\left(\frac{1}{\sqrt{d}}\mathsf{Q}^\top\mathsf{K} + \mathsf{M}\right), \quad \text{where} \quad \mathsf{M} = \begin{pmatrix} -\infty & 0 & 0 & \dots & 0 \\ 0 & -\infty & 0 & \dots & 0 \\ 0 & 0 & -\infty & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -\infty \end{pmatrix}. \tag{7.1}$$

We will now examine how this masking strategy affects the prediction of the last value in a sequence of N values, w.l.o.g we will use the notation $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N-1}, \boldsymbol{x}_N\}$ for this sequence. Following the background theory given in Section 6.2, each of the items has different projections (query, key, and values), i.e.,

$$\mathsf{Q} = [\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N)}], \quad \mathsf{K} = [\boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(N)}], \quad \mathsf{V} = [\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(N)}].$$

Using only one head for simplicity, the final representation for the N-th item we are interested to predict is obtained with:

$$\boldsymbol{z}_N = A_{1,N} \cdot \boldsymbol{v}^{(1)} + A_{2,N} \cdot \boldsymbol{v}^{(2)} + \ldots + A_{N-1,N} \cdot \boldsymbol{v}^{(N-1)} + 0 \cdot \boldsymbol{v}^{(N)}. \quad (7.2)$$

Despite the fact that the element $A_{N,N}$ equals zero and the value $\boldsymbol{v}^N$ is not explicitly included in the final representation $\boldsymbol{z}_N$, the definition of self-attention indirectly incorporates the contribution of the N-th element $\boldsymbol{x}_N$ in the representation. This is due to computing the attention values $A_{t,N}$ as the cosine-similarity of the items $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1}$ with respect to $\boldsymbol{x}_N$, as defined in Eq (6.7).

In this work, we aim to propose a strategy for computing attention values by utilizing the context of each item $\boldsymbol{x}_t$ rather than solely relying on their projected query and key values.

7.3 Unsupervised Diagnostic Spatio Temporal Transformer

In this section, we present the algorithm proposed in this study, which is referred to as the Unsupervised Diagnostic Spatio-Temporal Transformer. Additionally, we provide a detailed explanation of the proposed multi-masking strategy and anomaly score.

7.3.1 Overall architecture

The global architecture of uDFSTrans comprises three blocks, as illustrated in Figure 7.1. The first block employs multi-head 1DCNNs to extract embeddings from the input data, while preserving its temporal and spatial characteristics. The second block consists of two transformers, where the first transformer updates the embeddings based on the context to which each embedding belongs by applying different length maskings, and the second transformer learns the spatiotemporal relationships among these embeddings. Finally, the output layer is responsible for obtaining the reconstructions of the input data. Next, we provide a detailed explanation of the functioning of each block in uDFSTrans.

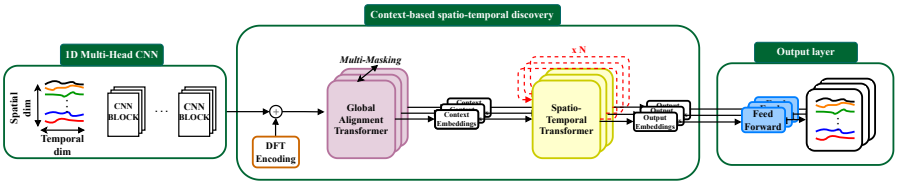


Fig. 7.1: Overall architecture of the proposed Unsupervised Diagnostic Spatio Temporal Transformer (uDFSTrans).

Multi-Head 1DCNN

The goal of automatically capturing informative ST patterns requires the capability of handling different temporal resolutions as well as learning ST dependencies.

As in our previous work, we employ a multi-head one-dimensional (1D) convolutional neural network (CNN), as the one used in [234], as illustrated in Fig. 6.3. In this case the input is a sequence of N ordered windows $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ that are mapped into a sequence of N ordered feature vectors $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.

After obtaining these features from the first block of uDFSTrans, it is necessary to introduce information about the order of these features before applying the transformers. To accomplish this, uDFSTrans employs the **faithfull encoding** described in Section 6.3.2

DFT encoding

As in the previous work, uDFSTrans also uses the **faithful-Encoding** to introduce the information about the positions of the features extracted. This information about the location is directly added to the features extracted by the Multi-Head 1D CNNs (see Section 6.3.2 and 6.4 for more details), i.e.,

$$\boldsymbol{f}_t \leftarrow \boldsymbol{f}_t + \boldsymbol{e}_t \tag{7.3}$$

where $\boldsymbol{e}_t$ is

$$\boldsymbol{e}_t = \sqrt{\frac{2}{d}} \left(\frac{1}{\sqrt{2}}, \cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_K t), \sin(\omega_K t), \frac{\cos \pi t}{\sqrt{2}} \right)^\top \tag{7.4}$$

Gobal alignment transformer

After introducing positional information to the feature vectors $\mathsf{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$, as in the original formulation of Vaswani et al. [28] the feature vectors are projected into value vectors using H different sets of parameter matrices $\{\mathsf{W}_V^{[h]} \mid h = 1, \ldots, H\}$. For simplicity, we will consider one head, i.e., $\mathsf{V} \triangleq \mathsf{W}_V \mathsf{F}$.

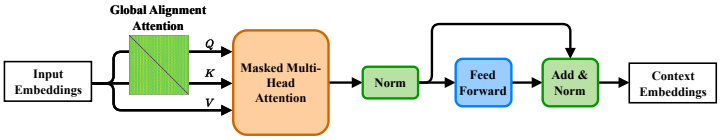


Fig. 7.2: Global Alignment Transformer.

Now, the goal is to alleviate the problem discussed in Sec. 7.2 about the original masking strategy when using transformers in forecasting. To update each value item $\boldsymbol{v}^{(t)}$ without implicitly using the query $\boldsymbol{q}^{(t)}$ and key $\boldsymbol{k}^{(t)}$ values corresponding to the same position, we propose the use of what we call a **Global Alignment Transformer** (GAT) (see Figure 7.2 for a detailed diagram). To achieve this, we propose obtaining representations of the query and key values based on the context to which an item in the sequence belongs. First, having a sequence of N items represented by features $\mathsf{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$, we define a fully learnable $N \times N$ attention matrix, called **Global Alignment Attention**, that will find typical connections between the features of the items in the sequence depending on the context. This learnable attention matrix is also composed of H different heads that allow learning more complex connections (we consider the case $H = 1$ for simplicity). The global alignment attention is defined as

$$\mathsf{A}_{GA} = \text{softmax}\left(\mathsf{W}_{GA} + \mathsf{M}^k\right) \tag{7.5}$$

where $\mathsf{W}_{GA} \in \mathbb{R}^{N \times N}$ is a learnable matrix and

$$\mathsf{M}^k_{i,j} = \begin{cases} -\infty, & \text{if} \quad i = j \quad \text{or} \quad |i-j| < k \\ 0, & \text{otherwise} \end{cases}$$

is a mask that for a given t position in a sequence will mask the features $\boldsymbol{f}_t$ and k neighbors around this position. Then, we generate contextualized vectors $\mathsf{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ through a matrix multiplication operation between the feature vectors F and A_{GA}, i.e.,

$$\mathsf{C} = \mathsf{F}\mathsf{A}^\top_{GA} \tag{7.6}$$

Thus, for a specific index t and considering the case where $k = 0$ as an illustrative example, the expression for the vector $\boldsymbol{c}_t$ is defined as follows:

$$\boldsymbol{c}_t = A^{(1,t)}_{GA} \cdot \boldsymbol{f}_1 + \ldots + A^{(t-1,t)}_{GA} \cdot \boldsymbol{f}_{t-1} + 0 \cdot \boldsymbol{f}_t + \tag{7.7}$$

$$A^{(t+1,t)}_{GA} \cdot \boldsymbol{f}_{t+1} + \ldots + A^{(N,t)}_{GA} \cdot \boldsymbol{f}_N. \tag{7.8}$$

While vector $\boldsymbol{c}_t$ holds relevant information about the sequence features, vector f_t is entirely masked. The query and key values are then calculated based on the contextual information provided by the $\boldsymbol{c}_t$ vectors as:

$$\mathsf{Q} \triangleq \mathsf{W}_Q\mathsf{C},\ \mathsf{K} \triangleq \mathsf{W}_K\mathsf{C} \tag{7.9}$$

Our underlying assumption is that the cosine similarity between the context-based query and key values can be employed to calculate the attention for the values V. In particular, when two distinct positions, denoted as $t1$ and $t2$, exhibit similar contextual representations ($\boldsymbol{c}_{t1}$ and $\boldsymbol{c}_{t2}$, respectively), the associated value vector $\boldsymbol{v}_{t2}$ can be utilized to accurately predict the output corresponding to the value vector $\boldsymbol{v}_{t1}$, and vice versa. Therefore, the values are updated as:

$$\mathsf{Z} = \mathsf{V}\mathsf{A}_C^\top, \quad \text{where} \quad \mathsf{A}_C \triangleq \text{softmax}\left(\frac{1}{\sqrt{d}}\mathsf{Q}^\top\mathsf{K} + \mathsf{M}^k\right). \tag{7.10}$$

In this way, the attention values are computed by taking into account the context of each individual item while simultaneously masking each of them.

After this, we apply layer normalization followed by two feedforward layers to the enriched representation Z:

$$\mathsf{Z} \leftarrow \text{LayerNorm}\Big(\text{LayerNorm}(\text{Dropout}(\mathsf{Z})) + \tag{7.11}$$

$$\text{Dropout}(\text{ReLU}\left(\mathsf{Z}\mathsf{W}_1 + b_1\right)\mathsf{W}_2 + b_2\Big) \tag{7.12}$$

where $\mathsf{W}_1, \mathsf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ and $b_1, b_2 \in \mathbb{R}^{d_{ff}}$ are the weights and biases of the first and second feedforward layers, respectively.

Spatio-temporal transformer

As multivariate time-series are considered in this study, these operations are employed for each sensor individually to obtain enriched sensor-wise context vectors. As a result, for each sensor indexed as $s \in \{1, \ldots, S\}$, a representation $\mathsf{Z}^{(s)} = \left\{\mathsf{Z}^{(1,s)}, \ldots, \mathsf{Z}^{(N,s)}\right\}$ is obtained from the corresponding context vectors $\mathsf{C}^{(s)} = \left\{\mathsf{C}^{(1,s)}, \ldots, \mathsf{C}^{(N,s)}\right\}$. Now, to model the spatio-temporal dependencies that may arise between these representations, we propose the use of a spatio-temporal transformer, as the spatio-temporal (ST) dependency discovery framework proposed in our previous work [244]. This spatio-temporal transformer is illustrated in Fig. 7.3.

To reflect ST dependencies in the representation vectors Z, we define the transition probability between the items assuming that there is a latent dependency structure behind $\{\mathsf{Z}^{(t,s)}\}$ described by the transition probability between two spatio-temporal points (t, s) and (t', s') as $p(t, s \mid t', s')$. Here we employ a factorized transition model

$$p(t, s \mid t', s') \approx p(t \mid t', s)p(s \mid s', t), \tag{7.13}$$

where $p(\cdot)$ is used as a symbolic notation representing probability distribution in general rather than a specific functional form.

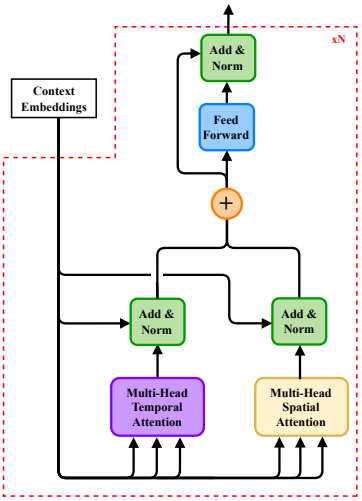


Fig. 7.3: Spatio-temporal dependency discovery framework.

Naming $\overline{\mathsf{Z}}$, $\hat{\mathsf{Z}}$ the spatial and temporal branches of Z, these transition probabilities are learned as

$$\hat{\mathsf{Z}}^{(s)} \leftarrow \underbrace{\text{softmax}\left(\frac{1}{\sqrt{d}}\hat{\mathsf{Z}}^{(s)}\hat{\mathsf{Z}}^{(s)\top}\right)}_{\textbf{Temporal attention:}\quad \mathsf{A}^{(s)}}\hat{\mathsf{Z}}^{(s)} \tag{7.14}$$

$$\overline{\mathsf{Z}}^{(t)} \leftarrow \underbrace{\text{softmax}\left(\frac{1}{\sqrt{d}}\overline{\mathsf{Z}}^{(t)}\overline{\mathsf{Z}}^{(t)\top}\right)}_{\textbf{Spatial attention:}\quad \mathsf{B}^{(t)}}\overline{\mathsf{Z}}^{(t)} \tag{7.15}$$

where $\hat{\mathsf{Z}}^{(s)} \triangleq [\mathsf{Z}^{(1,s)}, \ldots, \mathsf{Z}^{(N,s)}]$ and $\overline{\mathsf{Z}}^{(t)} \triangleq [\mathsf{Z}^{(t,1)}, \ldots, \mathsf{Z}^{(t,S)}]$. After this, the enriched representations are added, normalized, and represented in a single representation vector as

$$\mathsf{Z} \leftarrow \text{LayerNorm}(\mathsf{Z} + \text{Dropout}(\hat{\mathsf{Z}})) + \text{LayerNorm}(\mathsf{Z} + \text{Dropout}(\overline{\mathsf{Z}})). \tag{7.16}$$

Note that, for simplicity, although one head is used in Eq. (7.16), this operation is performed using H heads. After the aggregation, the enriched representation Z is normalized after being passed to a 2-layer feedforward network with ReLU activation function and a dropout layer:

$$\mathsf{Z} \leftarrow \text{LayerNorm}(\mathsf{Z} + \text{Dropout}(\text{ReLU}\left((\mathsf{Z}\mathsf{W}_1 + b_1)\,\mathsf{W}_2 + b_2\right))) \tag{7.17}$$

where $\mathsf{W}_1, \mathsf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ and $b_1, b_2 \in \mathbb{R}^{d_{ff}}$ are the weights and biases of the first and second feedforward layers, respectively.

Classification Head

Finally, the output layer utilizes a separate feedforward network for each sensor, with a sigmoid activation function, to derive the model's output, i.e.,

$$\mathsf{O}^{(N,s)} \leftarrow \text{sigmoid}(\mathsf{W}_O^{(s)}\mathsf{Z}^{(N,s)} + b_O^{(s)}) \tag{7.18}$$

where $\mathsf{W}^{(s)} \in \mathbb{R}^{d\times K}$ and $b_O^{(s)} \in \mathbb{R}^K$ are the learnable weights and the biases of the feedforward network.

Multi-masking Strategy

As depicted in the overall architecture of Fig. 7.1, three distinct branches can be observed in the second block of uDFSTrans. These branches apply different maskings to the features $\boldsymbol{f}_1, ..., \boldsymbol{f}_N$. Our hypothesis here is that if there is an anomaly at a certain window W_t, predicting the values of this window will become more challenging with an increased level of applied masking.

To measure this, we apply different maskings and then observe the differences between the predictions generated by each branch. In the two previous sections, we have elaborated on the workings of the Global Alignment Transformer, the ST transformer, and the output layer, which are abbreviated as GAT, STrans, and OutLayer, respectively. As depicted in Eq. (7.5), in GAT we add a mask M_k when calculating the attention, where k denotes the number of windows masked around a given temporal position t. In our experiments, we employ three branches with masking values of $k = 0, 1$, and 2 respectively. Each branch applies distinct masking lengths resulting in three distinct outputs, one from each masking approach:

$$\mathsf{Z}^{(\mathsf{k})} \leftarrow \text{GAT}^{(k)}(\mathsf{F}) \tag{7.19}$$

$$\mathsf{Z}^{(\mathsf{k})} \leftarrow \text{STrans}^{(k)}(\mathsf{Z}^{(\mathsf{k})}) \tag{7.20}$$

$$\mathsf{O}^{(\mathsf{k})} \leftarrow \text{OutLayer}^{(k)}(\mathsf{Z}^{(\mathsf{k})}) \tag{7.21}$$

In this way, we obtain three outputs $\left\{\mathsf{O}^{(0)}, \mathsf{O}^{(1)}, \mathsf{O}^{(2)}\right\}$ which will be the predictions for a given input window W_t masking $\{\mathsf{W}_t\}$, $\{\mathsf{W}_{t-1}, \mathsf{W}_t\}$ and $\{\mathsf{W}_{t-2}, \mathsf{W}_{t-1}, \mathsf{W}_t\}$, respectively.

Our supposition is that the dissimilarities observed among the three outputs can offer critical insights into identifying anomalies. Therefore, we propose an anomaly score that incorporates these differences, in addition to other pertinent factors.

7.3.2 Anomaly score

Below we define the terms that we take into account to establish the anomaly score.

- **Reconstruction error:**

$$\mathrm{RE_{t,s}} = \frac{\sum_{k=0}^{K} \|\mathsf{W}_{t,s} - \mathsf{O}_{t,s}^{(k)}\|}{\|\mathsf{W}_{t,s}\| + \epsilon}$$

  where K denotes the number of branches and $\mathsf{O}_{t,s}^{(k)}$ is the output for branch k and $\epsilon = 10e^{-4}$.
  If a window is anomalous, its reconstruction becomes more challenging. This is attributed to the model's training exclusively on 'normal' data, which does not encompass these outlier characteristics. Therefore, the reconstruction error will be higher in anomalous windows.
- **Inverse entropy of reconstruction errors:** Let $\mathbf{e}_{t,s} = \{\|\mathsf{W}_{t,s} - \mathsf{O}_{t,s}^{(k)}\| \quad | \quad k = 0, 1, 2\}$. Then, for time-step t and sensor s, the inverse entropy of reconstruction errors is defined as:

$$\mathrm{IHR_{t,s}} = \mathrm{H}(\gamma(e_{t,s}))^{-1}$$

  where $\mathrm{H}(X) := -\sum_{x \in \mathcal{X}} p(x) \log p(x) = \mathbb{E}[-\log p(X)]$ denotes the entropy function and $\gamma(e_{t,s}) = \frac{e_{t,s}}{\sum_{k=0}^{K} e_t}$.
  Increasing the number of masked windows complicates obtaining accurate reconstructions. Particularly, in anomalous segments the discrepancy in reconstructions is bigger with more extensive masking. Consequently, entropy in reconstruction errors inversely correlates with the noticeability of these differences, leading the inverse entropy function to yield elevated values when such differences are pronounced.
- **Attention variance between maskings:** Let $\mathsf{A}^{(l)} = \{\mathsf{A}^{(k,l)} \quad | \quad k = 0, 1, 2\}$ and $\mathsf{B}^{(k)} = \{\mathsf{B}^{(k,l)} \quad | \quad k = 0, 1, 2\}$ where K denotes the number of maskings used.

$$\begin{aligned}\mathrm{AV_{t,s}^{(k)}} = &\sum_{i=1}^{N} \left(\sum_{l=1}^{L} \mathrm{Var}(\mathsf{A}^{(l)}) \right)_{i,t} + \sum_{i=1}^{N} \left(\sum_{l=1}^{L} \mathrm{Var}(\mathsf{A}^{(l)}) \right)_{t,i} \\ &+ \sum_{j=1}^{N} \left(\sum_{l=1}^{L} \mathrm{Var}(\mathsf{B}^{(l)}) \right)_{j,s} + \sum_{j=1}^{N} \left(\sum_{l=1}^{L} \mathrm{Var}(\mathsf{B}^{(l)}) \right)_{s,j}.\end{aligned}$$

  Assuming that anomalous regions are more challenging to predict, the variance between the attention matrices produced by different masks could be larger in these areas. This is because there may be regions that are visible with one mask but not another, and these regions can be pertinent for prediction.
- **Attention variance between layers:** Let $\mathsf{A}^{(k)} = \{\mathsf{A}^{(k,l)} \quad | \quad l = 1, ..., L\}$ and $\mathsf{B}^{(k)} = \{\mathsf{B}^{(k)} \quad | \quad l = 1, ..., L\}$, where $\mathsf{A}^{(k,l)}$ and $\mathsf{B}^{(k,l)}$ denotes the temporal and spatial attention matrices for branch with masking k for layer l, respectively, and L denotes the number of transformer layers.

$$\mathrm{AV}^{(\mathrm{l})}_{\mathrm{t,s}} = \sum_{i=1}^{N} \left(\sum_{k=0}^{K} \mathrm{Var}(\mathsf{A}^{(k)}) \right)_{i,t} + \sum_{i=1}^{N} \left(\sum_{k=0}^{K} \mathrm{Var}(\mathsf{A}^{(k)} \right)_{t,i} + \sum_{j=1}^{S} \left(\sum_{k=0}^{K} \mathrm{Var}(\mathsf{B}^{(k)}) \right)_{j,s} + \sum_{j=1}^{S} \left(\sum_{k=0}^{K} \mathrm{Var}(\mathsf{B}^{(k)} \right)_{s,j} .$$

Under the same assumption, the connections within each masking branch will vary when regions display anomalous patterns. Therefore, the variance between the attention matrices across different layers will be higher when the regions are anomalous.

Considering these terms, the score for a given time-step t and a given sensor s is computed as:

$$\mathsf{S}_{t,s} = \mathrm{RE}_{\mathrm{t,s}} \cdot \mathrm{IHR}_{\mathrm{t,s}} \cdot \mathrm{AV}^{(\mathrm{l})}_{\mathrm{t,s}} \cdot \mathrm{AV}^{(\mathrm{k})}_{\mathrm{t,s}} \tag{7.22}$$

7.4 Experimental framework

In this section, we provide a description of the algorithms used for evaluation, as well as the datasets employed. We also outline the metrics used and the adopted evaluation strategy.

7.4.1 Datasets and baselines

A. Baseline algorithms

In our experiments, we compare uDFSTrans with nine state-of-the-art models for multivariate time-series anomaly detection, including GDN [139], DAGMM [143]. USAD [145], TranAD [212], CAE-M [159], MSCRED [144]. MTAD-GAT [245], OmniAnomaly [185] and MAD-GAN [246]. For a general understanding of how these algorithms work refer to Section 2.1 We use hyperparameters of the baseline models as presented in their respective papers.

B. Datasets

For the evaluation, the following datasets have been utilized:

- *Numenta Anomaly Benchmark (NAB)* [247]: This dataset comprises a variety of real-world data traces, including temperature sensor readings, cloud machine CPU utilization, service request latencies, and taxi demands in New York City. However, certain datasets such as Rogue and Rogue Updown have been excluded due to inconsistencies in their labels. Type of data: Univariate.
- *HexagonML (UCR)* [248]: This is a comprehensive collection of time series data, sourced from diverse domains, curated to facilitate and benchmark research in time-series anomaly detection. Type of data: Univariate.

- *Synthetic* [144]: Synthetic data is constructed as a series of time points, mathematically formulated as follows:

$$\mathcal{S}(t) = \begin{cases} \underbrace{\sin}_{C1}\underbrace{[(t-t_0)/\omega]}_{C2} + \underbrace{\lambda \cdot \epsilon}_{C3}, & s_{\text{rand}} = 0 \\ \underbrace{\cos}_{C1}\underbrace{[(t-t_0)/\omega]}_{C2} + \underbrace{\lambda \cdot \epsilon}_{C3}, & s_{\text{rand}} = 1 \end{cases} \tag{7.23}$$

  In the formula, s_{rand} is a binary random seed. It encapsulates three aspects of multivariate time series: (a) temporal patterns simulated by a trigonometric function (C1); (b) various periodic cycles represented by a time delay $t_0 \in [50, 100]$ and a frequency $\omega \in [40, 50]$ (C2); and (c) data noise, simulated by random Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ scaled by a factor of $\lambda = 0.3$ (C3). Type of data: Multivariate.
- *Server Machine Dataset* [185]: This is a five-week long dataset consisting of stacked traces of resource utilizations from 28 machines in a compute cluster, capturing both normal and anomalous patterns to aid in anomaly detection research. Type of data: Multivariate.
- *Secure Water Treatment (SWaT)* [249]: This dataset contains observations from a water treatment plant, encompassing seven days of regular operations and four days of abnormal or irregular operations. Type of data: Univariate.
- *MIT-BIH Supraventricular Arrhythmia Database (MBA)* [250]: This dataset comprises electrocardiogram recordings from four patients, featuring multiple occurrences of two distinct types of anomalies: supraventricular contractions and premature heartbeats Type of data: Multivariate.

7.4.2 Evaluation criteria

In this section, we will discuss the metrics used to evaluate our models for anomaly detection and diagnosis.

Anomaly detection

As the scores defined in the previous section can have different ranges of values for each of the sensors, we first normalize each dimension to be between 0 and 1. For anomaly detection, the scores for each time-step t are the sum of the scores obtained for each of the dimensions.

$$\mathsf{S}_t = \sum_{s=1}^{S} \text{MinMaxScaler}(\mathsf{S}_{t,s}) \tag{7.24}$$

The evaluation criterion used in recent years often relies on a protocol known as point adjustment. This protocol adjusts predictions in the following manner:

A dataset may contain multiple anomaly segments lasting over a few time steps. We denote $\mathcal{A}$ as a set of M anomaly segments; that is, $\mathcal{A} = \{A_1, \ldots, A_M\}$, where $A_m = \{t_s^m, \ldots, t_e^m\}$; t_s^m and t_e^m denote the start and end times of A_m, respectively. Using the PA protocolm the predictions $\hat{y}_t$ are adjusted to 1 for all $t \in A_m$ if the anomaly score is higher than a predefined threshold δ at least once in A_m, i.e.,

$$\hat{y}_t = \begin{cases} 1, & \text{if } \mathsf{S}_t > \delta \\ & \text{or } t \in A_m \text{ and } \underset{t' \in A_m}{\exists} \mathsf{S}_{t'} > \delta \\ 0, & \text{otherwise.} \end{cases} \tag{7.25}$$

However, a recent study by Kim et al. [251] demonstrates that this evaluation method may not be reliable. It has been experimentally shown that even a random anomaly score can yield better results than state-of-the-art algorithms. Consequently, we have adopted their evaluation criteria, named PA%K protocol. The PA%K protocol mitigates the overestimation effect of the PA protocol by applying the PA protocol to the segments A_m only if the ratio of correctly detected anomalies in A_m to its length exceeds the PA%K threshold K, i.e.

$$\hat{y}_t = \begin{cases} 1, & \text{if } \mathsf{S}_t > \delta \\ & \text{or } t \in A_m \text{ and } \underset{t' \in A_m}{\exists} \frac{\left|\left\{t' | t' \in A_m, \mathsf{S}_{t'} > \delta\right\}\right|}{|A_m|} > \mathrm{K} \\ 0, & \text{otherwise.} \end{cases} \tag{7.26}$$

Anomaly diagnosis

In terms of anomaly diagnosis, we have evaluated it on datasets where label information is also provided in the spatial dimension, specifically on the Synthetic and SMD datasets. In these datasets, for time step t the label $\boldsymbol{y}_t$ is given as $\boldsymbol{y}_t = \{\boldsymbol{y}_t^1, ..., \boldsymbol{y}_t^S\}$, where $\boldsymbol{y}_t^j \in \{0, 1\}$ for $s \in \{1, ..., S\}$. Particularly,

$$\boldsymbol{y}_t^s = \begin{cases} 1 & \text{if } \boldsymbol{x}_t^s \text{ is anomalous,} \\ 0 & \text{otherwise.} \end{cases}.$$

We employ standard metrics to assess the diagnostic efficiency of all models. HitRate@P% indicates the proportion of true anomaly dimensions the model successfully identified within its top predictions. The percentage P% corresponds to the proportion of anomalous dimensions in the ground truth at each timestamp, which we use to determine the top predicted candidates. For a detected anomaly $\boldsymbol{x_t}$, taking the anomaly score for each dimension and recording all of them into a list, AS_t, ordered by their contributions, let $\mathrm{GT}_t = \{s \quad | \quad s \in \{1, ..., S\} \wedge \boldsymbol{y}_t^s = 1\}$ be the ground truth array containing the dimensions contributing to the anomaly. Then, HitRate@P% is defined as:

$$\text{HitRate@P\%} = \frac{Hit@\lfloor P\% \times |GT_t| \rfloor}{|GT_t|} \tag{7.27}$$

where $|\text{GT}_t|$ is the length of GT_t. For example, at timestamp t, if 2 dimensions are flagged as anomalous in the ground truth, HitRate@100% would consider the top 2 dimensions, while HitRate@150% would account for 3 dimensions.

In addition, we calculate the Normalized Discounted Cumulative Gain (NDCG), where NDCG@P% considers an equivalent number of top predicted candidates as HitRate@P%.

7.5 Results and discussion

In this section, we aim to demonstrate the ability of uDFSTrans to detect and diagnose anomalies. To show this, we compare its performance across six datasets, utilizing nine different baseline methods for reference.

7.5.1 Anomaly detection

First, we start by checking how well the model can spot anomalies. For the evaluation, we use the PA%K evaluation protocol to measure the F1 and AUC scores for different k values.

The results we obtained from each dataset are visualized in Figure 7.4, where we have plotted the model's performance trend as the percentage of k increases. The F1 score is depicted with dots, while the AUC is represented with crosses. We have used a different color for the lines connecting the dots and crosses for each model. According to the PA%K evaluation protocol, the less the performance drops as k increases, the more reliable the anomaly scores provided by the model are.

Analyzing the results from Figure 7.4, we find that, on the whole, uDFSTrans is the model whose performance decreases the least as k increases. When evaluated in terms of AUC, we observe that uDFSTrans generally outperforms the other algorithms, delivering the best results in the Synthetic, NAB, UCR, and MBA datasets. It holds its own in the SMD dataset and faces more challenges in the SWaT dataset. However, when looking at the results for all values of k, it is evident that in the SWaT dataset, uDFSTrans is the model whose performance decreases least with larger k values, which makes the scores to be reliable. Looking at the F1 score, uDFSTrans also delivers very competitive results, ranking as the best in MBA, UCR, NAB, and Synthetic datasets and holding a competitive position in both SWaT and SMD datasets.

We have presented these results numerically in Table **??**, taking the average of the results obtained for the different k values. As seen in the bold-highlighted results, uDFSTrans achieves the best performance in most datasets in terms of F1 and AUC scores.

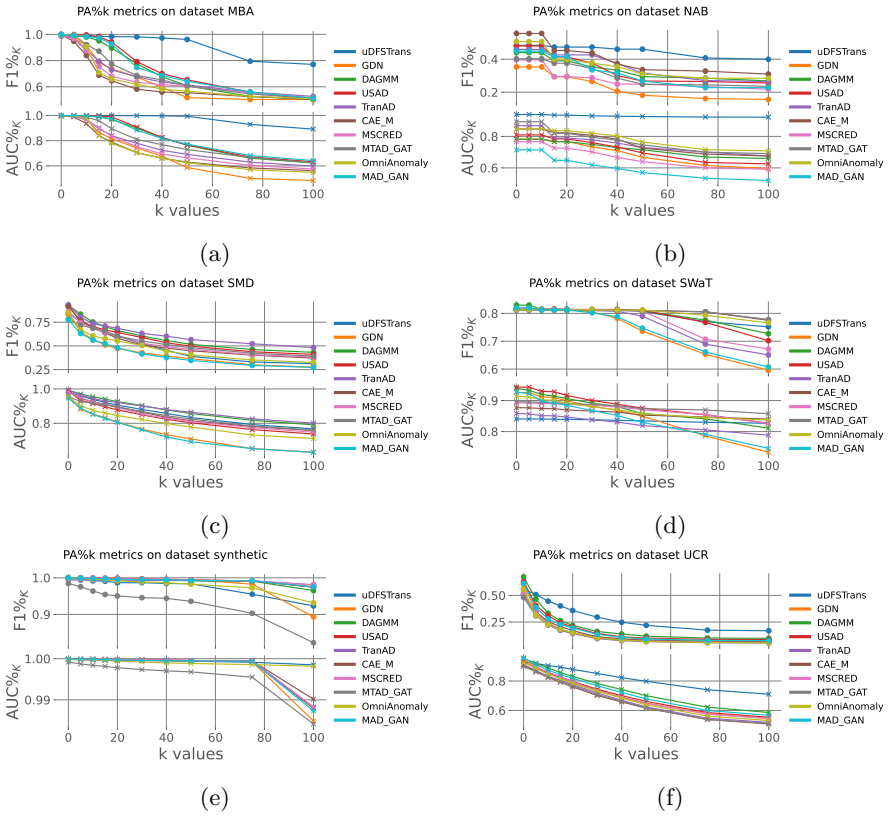


Fig. 7.4: Results obtained for each model in six different univariate and multivariate anomaly detection datasets using PA%$_k$ evaluation protocol for AUC and F1 scores.

In Figure 7.5, we have plotted some results obtained from the UCR dataset. In this figure, the true data is represented in black, the given prediction is in red (note that we have only plotted output O_1 to keep the image clear and avoid confusion), the ground truth label is in purple, the obtained anomaly scores are in green, and we have used yellow for the threshold and predictions. As seen in the image, the scores obtained are higher in anomalous areas, while in non-anomalous areas, we generally get lower scores. In all three cases, we see that the model has been successful in detecting the anomalies.

7.5.2 Anomaly diagnosis

For anomaly diagnosis, the evaluation has been done based on two metrics: Hit@k% and NDCG@k%, for k values of 100 and 150. The anomaly diagnosis has been evaluated in two datasets, which provide labels for each sensor as

Table 7.1: Anomaly detection results based on PA%K evaluation protocol. The results are the mean values of F1 and AUC scores obtained for different k percentages. The best results are highlighted in **bold**.

Model	Synthetic		NAB		UCR		SWaT		MBA		SMD		**Average**	
	$F1_k$	AUC_k	$F1_k$	AUC_k	$F1_k$	AUC_k	$F1_k$	AUC_k	$F1_k$	AUC_k	$F1_k$	AUC_k	$F1_k$	AUC_k
GDN	0.985	0.998	0.262	0.721	0.181	0.756	0.765	0.869	0.720	0.763	0.490	0.783	0.567	0.815
USAD	0.995	0.999	0.373	0.745	**0.212**	0.760	0.798	**0.900**	0.813	**0.878**	0.616	0.859	0.634	0.840
TranAD	0.992	0.998	0.388	0.783	0.167	0.729	0.782	0.834	0.739	0.812	0.667	0.902	0.622	0.843
CAE-M	0.995	0.999	0.436	0.787	0.190	0.717	**0.808**	0.864	0.684	0.769	0.580	0.872	0.616	0.821
MSCRED	**0.996**	0.999	0.301	0.695	0.174	0.746	0.788	0.877	0.720	0.799	0.582	0.867	0.593	0.830
MTAD-GAT	0.938	0.996	0.332	0.803	0.161	0.717	0.807	0.884	0.723	0.814	0.590	0.874	0.592	0.848
MAD-GAN	0.980	0.999	0.356	0.628	0.202	0.773	0.768	0.861	0.802	0.876	0.468	0.778	0.596	0.819
DAGMM	0.993	0.999	0.364	0.740	0.243	0.789	0.804	0.889	0.791	0.872	**0.645**	**0.903**	0.625	0.842
OmniAnomaly	0.984	**1**	0.393	0.803	0.169	0.745	0.806	0.884	0.707	0.770	0.528	0.828	0.598	0.821
uDFSTrans (Ours)	0.980	**1**	**0.460**	**0.932**	**0.338**	**0.845**	0.803	0.837	**0.940**	**0.980**	0.556	0.884	**0.680**	**0.896**

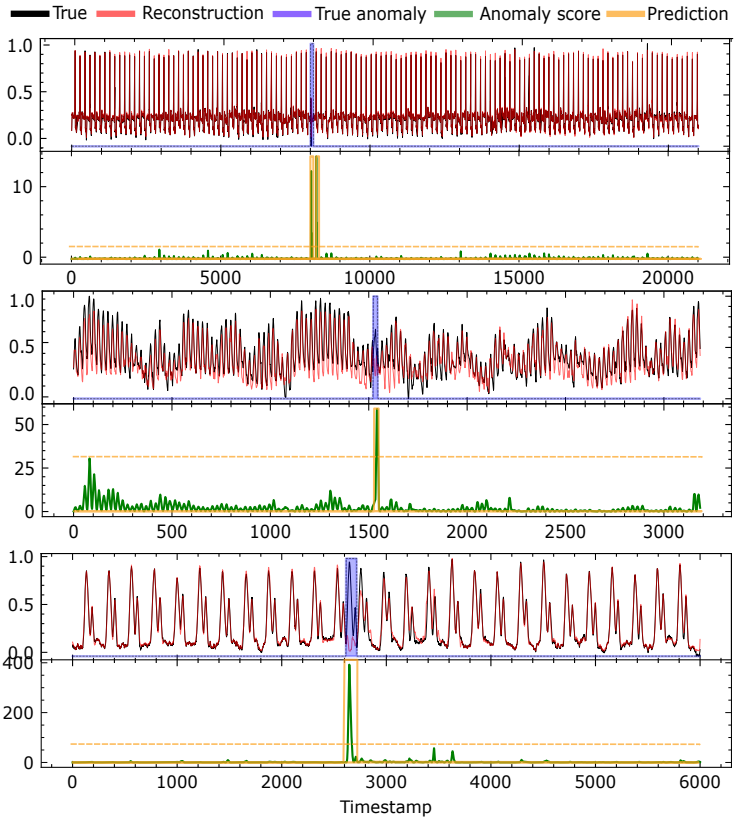


Fig. 7.5: Examples of uDFSTrans results in three UCR datasets.

well as temporal labels, namely the SMD and Synthetic datasets. The results can be viewed in Table 7.2.

Table 7.2: Anomaly diagnosis results for SMD and synthetic dataset based on Hit@k% and NDCG@k% values. The best results are highlighted in **bold**.

Model	Hit@100%		Hit@150%		NDCG@100%		NDCG@150%	
	SMD	Synthetic	SMD	Synthetic	SMD	Synthetic	SMD	Synthetic
GDN	0.308	0.808	0.439	0.853	0.315	0.831	0.393	0.856
USAD	0.452	0.987	0.561	0.989	0.467	0.987	0.533	0.988
TranAD	0.458	0.986	0.558	0.990	0.483	0.987	0.543	0.989
CAE-M	0.414	0.946	0.527	0.972	0.435	0.959	0.502	0.974
MSCRED	0.434	0.830	0.555	0.894	0.454	0.869	0.527	0.906
MTAD-GAT	0.354	0.826	0.474	0.918	0.377	0.851	0.449	0.904
MAD-GAN	0.402	0.963	0.533	0.986	0.392	0.971	0.470	0.985
DAGMM	**0.512**	0.987	0.626	0.989	**0.526**	0.987	0.595	0.988
OmniAnomaly	0.433	0.985	0.534	0.995	0.447	0.989	0.507	0.995
uDFSTrans (ours)	0.506	**0.999**	**0.650**	**1.000**	**0.526**	**0.999**	**0.612**	**1.000**

Analyzing these results, we can say that uDFSTrans outperforms most of the other algorithms for anomaly diagnosis in both datasets. Based on the Hit@100% metric, only the DAGMM algorithm achieves better results than uDFSTrans in the SMD dataset, but in all other cases, uDFSTrans performs better in all terms. For the SMD dataset, it is worth mentioning that these two algorithms outperform all the other baseline methods.

We also find it interesting to visualize how the terms of the scores are affected when there is an anomaly in the model's input. To this end, in Figure 7.6, we perform a visual diagnosis of an anomaly by plotting an input with an anomaly at the end of the sequence and simultaneously plotting the variance matrix of the attention mechanisms to observe their behavior in the anomalous windows. In plot (c) of this figure, we can see that these variances are higher in the region where the anomaly occurs. This is reflected in the plots of figure (d), specifically where it is visualized that the attention variance between different layers and the attention variance between different maskings is higher in anomalous regions. On one hand, this phenomenon can be attributed to the maskings, as there are regions that are visible with one masking but not with another, and these regions can be valid for prediction. On the other hand, it's due to the fact that the connections of the attention mechanism change when a region exhibits abnormal behavior.

Moreover, we have plotted also the inverse entropy of reconstruction errors. As observed in the reconstructions, as the masking increases, the reconstruction worsens, leading to an increase in reconstruction error. This phenomenon is more noticeable in anomalous regions as they are harder to reconstruct, given that only instances with normal behavior were used in training. There-

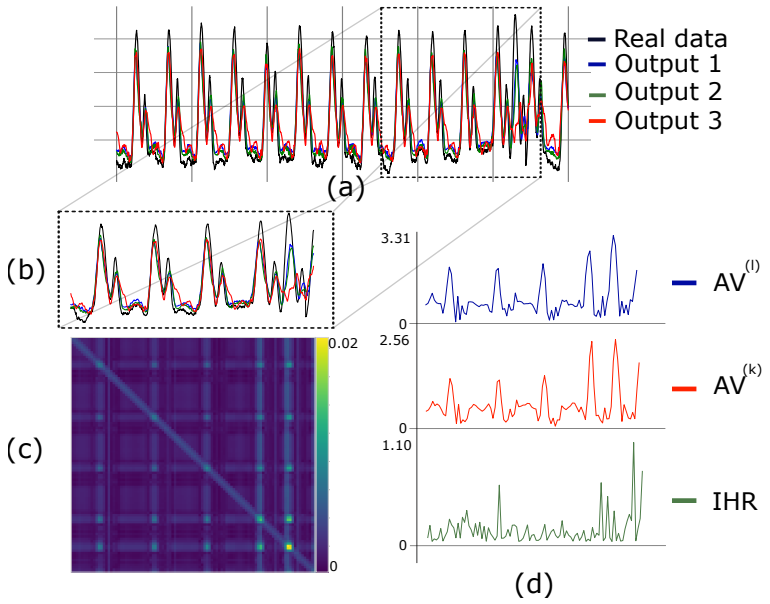


Fig. 7.6: Visual diagnostics of an anomalous subsequence. **(a)** shows a time-series with the reconstructions obtained using different masks, **(b)** shows a zoomed area of the time-series to be analyzed, **(c)** shows the variance matrix of all the attention matrices, and **(d)** shows, on top, the score obtained from the attention variances between different transformer layers (denoted as $\text{AV}^{(l)}$), in the middle, the score obtained from the attention variances between different masking branches (denoted as $\text{AV}^{(k)}$), and at the bottom, the inverse entropy of the reconstruction errors.

fore, in the last plot of Figure 7.6, we note a variation in reconstruction errors, measured by the inverse entropy. As the disparity in these errors grows in the anomalous regions, the entropy reduces, leading to higher IHR values in such areas.

7.6 uDFSTrans, a reliable solution for unsupervised anomaly detection and diagnosis

With this research, we have written an article that has been submitted to the journal IEEE Transactions on Pattern Analysis and Machine Intelligence

Article 4 (A4): Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Transformers are Efficient Unsupervised Anomaly Detectors*. IEEE Transactions on Pattern Analysis and Machine Intelligence. **Status**: *Under review*

In this study, we proposed a spatio-temporal transformer-based algorithm that utilizes a multi-masking strategy and context-based attention mechanism, which allows for the efficient detection and diagnosis of anomalies in multivariate time series in an unsupervised fashion. With this, we have demonstrated that transformers are effective tools for anomaly detection, and that the spatio-temporal relationships they learn contribute to enhancing the diagnosis of anomalies.

Our evaluation shows that the suggested model, uDFSTrans, performs consistently well in detecting anomalies. This consistency is confirmed by the stable results in F1 and AUC scores across different k values in the PA%K assessment protocol. This is indicative of its ability to provide reliable anomaly scores. The proposed anomaly score effectively brings together key factors to distinguish anomalies. On one hand, the use of a multi-masking strategy enables us to observe greater differences in reconstruction errors, which in turn facilitates anomaly differentiation using inverse entropy. On the other hand, we have demonstrated that variance within attention mechanisms tends to be greater in anomalous regions, which is a crucial factor in detecting these regions.

In terms of anomaly diagnosis, the results obtained affirm the efficacy of uDFSTrans in diagnosing anomalies. The proposed algorithm outperforms the baseline algorithms in the metrics HIT@k% and NDCG@k% scores. Therefore, we conclude that, in addition to being effective in detecting anomalies in the temporal space, the spatial connections also help in detecting anomalies in the spatial dimension, which is crucial for anomaly diagnosis.

Part III

General Conclusions and Future Work

8

General Conclusions and Future Work

Finally, in this section, we summarize the general conclusions derived from this thesis and highlight potential future work.

8.1 General conclusions

Throughout this research, we have made significant strides towards our overarching objective (**GO**) of **researching, designing, and validating explainable solutions for anomaly diagnosis in time-series data**. Our research has primarily focused on two areas: post-hoc real-time explainability for time-series anomaly detection and integrating intrinsic interpretability in black-box models. In addressing these objectives, we have made four key contributions.

To fulfill this goal, our initial steps were directed towards embracing post-hoc techniques to provide real-time interpretability for such data. It was within this realm that we identified counterfactual explanations as an area deserving of our keen attention. Our exploration is carefully structured to address our first hypothesis (**H1**): **post-hoc XAI methods enable real-time interpretation of time series data, providing anomaly diagnosis.**. In alignment with this hypothesis, our first major goal, **O1**, propels the first study of the thesis: **to research, design and validate post-hoc real-time explainability for time-series anomaly detection.**

In the realm of anomaly detection, swift and effective responses are imperative. Counterfactual explanations emerge as a powerful tool in this scenario. **They not only pinpoint the root of detected anomalies but also provide crucial guidance on subsequent rectification steps**, enhancing the efficiency of both diagnosis and remediation. Thus, **our first contribution involved the validation of the use of the Counterfactual Explanations Method (CEM) in time-series classification tasks, establishing the applicability of CEM in this context and exploring its potential as a tool for generating meaningful explanations**. Despite the fact

that CEM was capable of producing counterfactuals in time-series data, **we noted the computational demands of CEM as a significant drawback**, which motivated us to address this in our subsequent exploration: conceiving a method optimized for real-time counterfactual explanations. This contribution led us to author the following article:

Article 1 (A1): Labaien, J., Zugasti, E., & De Carlos, X. (2020, September). *Contrastive explanations for a deep learning model on time-series data.* In Big Data Analytics and Knowledge Discovery: 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings (pp. 235-244). Cham: Springer International Publishing. **Status**: *Accepted.*

For our subsequent contribution, drawing inspiration from our initial findings and addressing the time constraints of CEM, **we introduced the Real-Time Guided Counterfactual Explanations (RTGCEx) method**. **This approach uses autoencoders, trained with a multi-objective loss function, to craft real-time counterfactual explanations.** Beyond merely being fast, RTGCEx is designed to be model-agnostic and emphasizes delivering user-focused explanations.

The true value of RTGCEx is seen in its ability to produce fast counterfactuals across various domains and types of data. Our results on the MNIST dataset show that it not only outperforms traditional techniques in speed but also excels in producing quality insights. Further tests on the Gearbox dataset spotlight also RTGCEx's proficiency in effectively detecting and addressing anomalies. With these experiments, we have written another article:

Article 2 (A2): Labaien Soto, J., Zugasti Uriguen, E., & De Carlos Garcia, X. (2023). *Real-Time, Model-Agnostic and User-Driven Counterfactual Explanations Using Autoencoders.* Applied Sciences, 13(5), 2912. **Status**: *Accepted.*

After our initial exploration of post-hoc techniques, we turned our attention to intrinsically interpretable models. Guided by our second hypothesis (**H2**): we believed that **the integration of interpretability within a black box model can improve the explainability without compromising performance metrics**. To achieve this, our second objective (**O2**) was to **research, design, and validate the integration of intrinsic interpretability in black-box models for time-series anomaly detection and diagnosis.**

In the previous works, our emphasis was on methods that elucidate the reasoning behind a model's conclusions, particularly concerning real-time interpretation of time-series data. However, it becomes apparent that **such explanations may not always encapsulate the nuanced details embedded**

within the data and the models. These elucidations, being detached from the intrinsic workings of the models, might omit essential information. This motivated us to directly **integrate interpretability within anomaly detectors, facilitating a more comprehensive and immediate understanding during the diagnostic process**. In this context, we believe that Transformer models are suitable for providing these explanations due to their ability to capture spatio-temporal relationships within the data.

The third contribution presents **a new framework for diagnosing anomalies in multivariate sensor data, called Diagnostic Fourier-based Spatio-temporal Transformer (DFSTrans)**. At its core, this framework blends the strengths of multi-head 1D CNNs, which capture detailed insights from raw data, with the benefits of a Transformer-like setup that understands how these insights connect over time.

In this research, **we found a challenge with the usual way Transformers understand the position of data.** This standard method sometimes misses short-term changes, which are often vital for spotting anomalies. To solve this, **we introduced a new approach, called faithful encoding. This method ensures that every piece of data is given equal importance, without any biases.**

We put our new DFSTrans framework to the test using four different sets of data. We also compared our new faithful encoding with the usual method. The results were promising: **our method often performed better, and our attention-focused tools proved reliable in diagnosing anomalies.** In simple terms, the DFSTrans framework is a reliable tool for spotting and understanding unusual patterns in complex data from sensors. With these experiments, we authored another article:

Article 3 (A3): Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Diagnostic Spatio Temporal Transfomer with Faithfull Encoding*. Knowledge-Based Systems. **Status**: *Accepted*.

Thus far, our research has predominantly resided in the supervised learning domain, which relies on the presence of labeled data. However, **obtaining such labels, particularly for anomalous data, can be challenging.** Given this obstacle, we have transitioned to refining our DFSTrans framework for unsupervised scenarios, enhancing its adaptability for diverse real-world applications.

As stated, our fourth contribution involves adapting our framework to an unsupervised setting. We introduced the unsupervised version of the DFSTrans framework (**uDFSTrans**), **which uses a multi-masking strategy and a context-based attention mechanism.** This adaptation proves good performance and consistency in anomaly detection and diagnosis in an unsupervised manner, thus enhancing its applicability in real-world scenarios where labeled anomalies may not be available.

In our tests, the uDFSTrans model consistently showed its strength in detecting anomalies. We obtained competitive results in both F1 and AUC scores when looking at various test values under the PA%K evaluation protocol. This ensures the model's ability to produce trustworthy scores for anomalies. This ability to detect anomalies hinges on a couple of key features. Firstly, using a multi-masking technique helps us notice significant differences when data is reconstructed, aiding us in differentiating between normal and anomalous patterns. Secondly, we found that attention tools show more variation when they are in the vicinity of anomalies, which is a factor that helps highlighting these unusual areas.

When it comes to diagnosing the nature and cause of these anomalies, uDFSTrans proves its worth again. In terms of HIT@k% and NDCG@k% scores, uDFSTrans delivered better results than other standard methods. This leads us to believe that uDFSTrans is not just adept at spotting time-based irregularities, but it's also skilled in understanding spatial anomalies, a key aspect of comprehensive anomaly detection.

With this research, we have authored the last article of the thesis, which has been submitted to the journal IEEE Transactions on Pattern Analysis and Machine Intelligence:

Article 4 (A4): Labaien, J., Ide T., Chen, P.Y, Zugasti, E., & De Carlos, X. (2023). *Transformers are Efficient Unsupervised Anomaly Detectors*. IEEE Transactions on Pattern Analysis and Machine Intelligence. **Status**: *Under review*

Overall, our research has successfully validated our general hypothesis: Explainable Artificial Intelligence methods can provide valuable insights into the decision-making process of black box models, leading to accurate anomaly diagnosis. Moreover, we have been successful in fulfilling our general objective of researching, designing, and validating explainable solutions for anomaly diagnosis in time-series data. Furthermore, our efforts align well with our specific objectives of providing real-time interpretation of time-series data and integrating interpretability within black box models without compromising their performance. The tools and frameworks we have developed not only improve anomaly detection and diagnosis but also enhance the understandability of these processes, leading to better, more informed decisions in various application domains.

Following the conclusions, it is worth noting that the final two publications emerged from my stay at the IBM Research Center in Yorktown Heights, New York, carried out from September 2022 to December 2022. Throughout this period, I had the privilege of working under the expert guidance of Pin-Yu Chen and Tsuyoshi Ide. Their insights and mentorship were instrumental in shaping these significant contributions.

8.2 Future work

In light of the findings and experiences from this research, several future directions arise. These potential lines of inquiry can further enhance our understanding of time-series anomaly detection and continue to push the boundaries of our knowledge.

- **Analysis of counterfactual explanations in multivariate time-series data:** This research opens up opportunities for expanding the understanding of counterfactual explanations in multivariate time-series data. Our study has demonstrated the utility of such explanations in univariate time-series data. Still, the complexity and dependencies within multivariate time-series data present new challenges and opportunities that can be explored.
- **Strategies for context connections in spatio-temporal settings:** Our work has utilized global alignment attention, which considers the temporal context sensor-wise separately. Future work could explore other strategies for forming context connections that take into account both spatial and temporal information in a more integrated manner.
- **Hyperparameter exploration in spatio-temporal transformers:** The effects of different hyperparameters on the performance of spatio-temporal transformers remain largely unexplored. Future studies could delve into this area, determining the optimal hyperparameters and how they impact anomaly detection and diagnosis.
- **Automatic threshold selection algorithms in anomaly score-based detectors:** The selection of a suitable threshold in anomaly score-based detectors is crucial, and this study has applied predefined thresholds. Future research could aim at developing and testing automatic threshold selection algorithms to improve the adaptability and performance of these detectors.
- **Normalization strategies for online settings:** As our work has primarily focused on batch processing, a promising future direction could be the exploration of normalization strategies suitable for online settings. This would allow our models to better adapt to streaming data and real-time applications.
- **Metrics for attention-based anomaly diagnosis:** Our research has utilized conventional metrics for evaluation. However, given the unique characteristics of attention-based anomaly diagnosis, the exploration and development of dedicated metrics could provide a more nuanced understanding of these models' performance.
- **Exploration of concept drift scenarios:** One of the common challenges in time-series data analysis is the phenomenon of concept drift, where the statistical properties of the target variable, which the model is trying to predict, change over time. This makes the patterns that the model learned at some point become obsolete or less effective over time. Future research

could focus on developing strategies for handling concept drift in anomaly detection in time-series data, ensuring the models remain effective as new data is collected over time.

Part IV

Appendix

9

DFSTrans

9.1 Elevator use-case

The industrial use case of this work deals with an elevator monitored by 20 sensors. The data used in this work is obtained from a physical model designed by a domain expert that mimics the real behavior of the elevator. This simulator is highly configurable, having different parameters to determine the power of the electric machine, the load of the cabin, the alignment of the guidance system, or the tension of the cables, among others. The physical model simulates up and down journeys under all possible conditions by modifying these input parameters. Specific values of these parameters allow generating fault conditions during the journey, such as reduced lubrication, misalignment or peaks in the guiding system, or de-magnetization or loss of inductance in the electric machine. This paper studies the most common issue in this case study: the failures produced in the guiding system. In order to generate these anomalies, as pointed out before, three effects have been introduced to the model related to misalignment, lubrication reduction, and localized bumps in the guiding system. The first two effects increase friction between the cabin, the counterweight, and the rails. The localized bumps result in local impacts on the cabin, leading to spike-like malformations. These effects cover the main fault conditions of the guiding systems. All failures can be observed in several of the described sensors, but they may not be reflected in all of them.

9.2 1D Multi-Head CNN

9.2.1 Architecture and hyperparameters

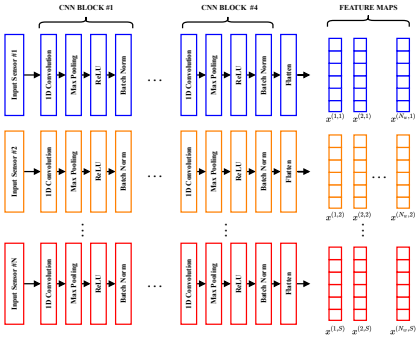


Fig. 9.1: 1D Multi-Head CNN used for feature extraction.

Table 9.1: Hyperparameters for Multi-Head 1D CNN.

Parameter	Datasets			
	Elevator	SMD	MSL	SMAP
Number of sensors (S)	20	38	55	25
Window length (w_l)	100	50	50	50
Time-segments (N_w)	80	10	10	10
Number of convolutional blocks	4	4	4	4
Kernel size	5	5	5	5
Embedding dim (d)	240	120	120	120
Pooling size	2	2	2	2
Pooling stride	2	2	2	2

Figure 9.1 shows the architecture of the 1D Multi-Head CNN used in DFStrans and Strans. The only difference concerning the blocks proposed by Cañizo et al.[234] is the introduction of Max Pooling layers. As shown in Figure 9.1, each convolution applies a Max Pooling layer, followed by a ReLU activation function and a Batch Normalization (BN) [252] layer. The Max Pooling layer helps to significantly reduce the dimensionality of the extracted features by statistically summarizing the output of the convolutional layers at a given time point based on their neighbors [127].

Table 9.1 shows the details of the data used, such as the number of sensors, the length of the windows or the number of time segments and also shows the hyperparameters used in each dataset for the 1D Multi-Head CNN. These are chosen based on the exhaustive analysis made by [234] in their paper.

9.2.2 Comparison with linear projection

Next, we want to measure the effect that the Multi-Head 1D CNN has on the performance of DFStrans in comparison to whether the embeddings that are fed to the spatio-temporal dependency discovery network are achieved by linear projections, as done in [233], and non-linear projection, as done in [228].

As stated, the MH 1D CNN maps each raw time-series t-segment to a representation matrix: $U_i^{(t)} \in \mathbb{R}^{S \times w_L} \rightarrow \mathsf{X}_i^{(t)} \in \mathbb{R}^{S \times M}$, where w_l denotes the window length and M is the dimensionality of the embedding. These embeddings are obtained by means of different convolutional blocks in our case. But in [233] for example, these embeddings are linear projections of the raw data, i.e.

$$\mathsf{X}_i^{(t)} \triangleq U_i^{(t)}\mathsf{W}^{(t)}, \tag{9.1}$$

or non-linear projections, as in [228], i.e.

$$\mathsf{X}_i^{(t)} \triangleq \tanh(U_i^{(t)}\mathsf{W}^{(t)}), \tag{9.2}$$

where $\mathsf{W}^{(t)} \in \mathbb{R}^{w_l \times M}$ is a fully learneable matrix.

Table 9.2: Comparison between DFStrans using Multi-Head 1DCNN and DFStrans with linearly and non-linearly projected embeddings.

Feature extraction	Precision	Recall	F1
Multi-Head 1D CNN	**0.989 ± 0.016**	**0.917 ± 0.022**	**0.952 ± 0.005**
Linear projection	0.986 ± 0.002	0.668 ± 0.045	0.778 ± 0.041
Non-linear projection	0.980 ± 0.002	0.85 ± 0.043	0.912 ± 0.030

Table 9.2 shows the results using different feature extraction strategies to embed the raw time-series. This experiment has been performed for the elevator use case. Looking at the results, we see that embedding extraction plays an essential role in detecting anomalies. Using Multi-Head 1D CNN as a feature extractor, we outperform the other strategies in terms of Recall and F1. Although the Precision is still high with the other methods, it loses detection capability, and the variability is higher too. Among the other two strategies, non-linear projections performed better than linear projections.

9.3 Visualization of positional encodings

In Figure 9.2 we have visualized, on the one hand, the vanilla positional encoding, and on the other hand, the faithful-Encoding that we propose. In this example, we have used $N_w = 80$ time segments and $d = 240$ features.

9.4 Details of spatio-temporal dependency structure and classification head

Table 9.3 shows the hyperparameters used in the proposed network. Although the experiments can be generalized to more heads and more layers in the spatio-temporal dependency structure, we focus on a single layer and a single attention head to facilitate the interpretation of the network for anomaly diagnosis. The number of units for the spatial and temporal attention feed-forward layer is 2048. The activations we use in the network are generally ReLUs, except for the last classification layer, which is a sigmoid. Moreover, the dropout rate is 0.1 in the whole network.

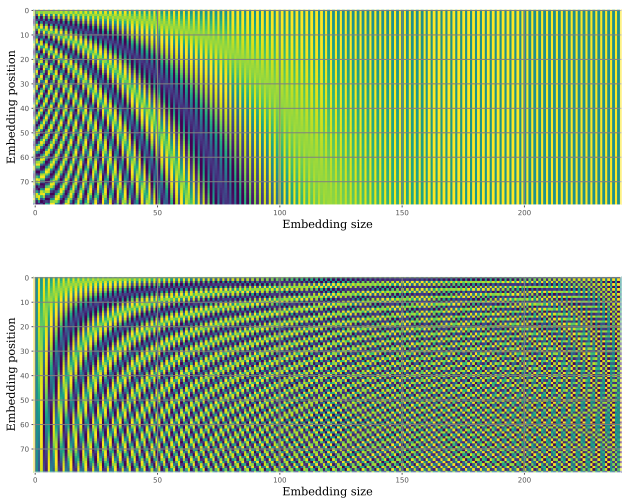


Fig. 9.2: The figure on **top** shows a visualization of vanilla positional encoding and the figure of the **bottom** shows a visualization of **faithful-Encoding**. Both for $N_w = 80$ and $d = 240$.

Table 9.3: The hyperparameters selected for the spatio-temporal dependency structure. Same selection for every dataset.

Hyperparameters		**Value**
ST dependency structure	*Dim feedforward*	2048
	Dropout rate	0.1
	Number of attention heads	10
	Number of layers	1
	Activations	ReLU
Classification head	*Dim feedforward 1*	512
	Activation in feedforward 1	ReLU
	Dropout rate	0.1
	Dim feedforward 2	1
	Activation in feedforward 2	Sigmoid

A. ***Justification of the transition probability equations***

These are the equations that we are going to justify:

$$p(t, s \mid t', s') \approx p(t \mid t', s)p(s \mid s', t), \tag{9.3}$$

and

$$\ln p(t \mid t', s) = \text{const.} + \frac{1}{\sqrt{M}}(\hat{\boldsymbol{x}}^{(t,s)})^\top \hat{\mathsf{H}} \hat{\boldsymbol{x}}^{(t',s)}, \tag{9.4}$$

$$\ln p(s \mid s', t) = \text{const.} + \frac{1}{\sqrt{M}}(\overline{\boldsymbol{x}}^{(t,s)})^\top \overline{\mathsf{H}} \overline{\boldsymbol{x}}^{(t,s')}, \tag{9.5}$$

The key motivation of Eq. 9.3 is how to take account of the dependency between spatial and temporal coordinates. The most naive approach is to treat them as independent like

$$p\left(t, s \mid t', s'\right) \approx p\left(t \mid t', s'\right) p\left(s \mid t', s'\right) \quad \textbf{(naive)} \tag{9.6}$$

However, this is not the best model because we know that different sensors (s, s') have different time correlations, and spatial dependency can vary at different times (t, t'). Thus, Eq.9.3 moves one step ahead of this naive model and can be viewed as a tractable but still tractable approximation of the full model. Instead of naively assuming independence, we used

$$p\left(t, s \mid t', s'\right) \approx p\left(t \mid t', s\right) p\left(s \mid t, s'\right) \quad \textbf{(ours)} \tag{9.7}$$

so that the dependency between s and t is captured to some extent. Moreover, the Eqs. 9.4 and 9.5 have two relatively clear justifications.

1. It leads to the well-known query-key formalism of the transformer.
2. Eqs.9.4 and 9.5 amount to approximating the distribution up to the second order moments. This is indeed a widely used technique. For instance, you can think of it as a Laplace approximation (see [253]) Section 8.4.1), where the idea is essentially "use the 2nd order Taylor expansion of $\ln p$".

9.5 Details of benchmark algorithms

A. Used code.

Here are the links to the codes for the benchmark datasets:

- **InceptionTime:** https://github.com/TheMrGhostman/InceptionTime-Pytorch
- **TapNet:** https://github.com/xuczhang/tapnet
- **MLSTM-FCN:** https://github.com/metra4ok/MLSTM-FCN-Pytorch
- **Multi-Head 1D CNN - LSTM:** No open source code has been found for this algorithm.

B. ***Hyperparameters***

Table 9.4 shows the hyperparameters selected in each benchmark algorithm for each dataset. In most cases the hyperparameters selected are the default parameters that the algorithms provide, but there have been some that we have considered changing. For example, in the case of TapNet, the *dilatation* and *rp params* parameters depend on the input dimension and the length of the series, so it varies for each case. The *rp param* consists of two parameters: the first one corresponds to the number of permutations, and the second one to the sub-dimension of each permutation. Following the official implementation, the number of permutations is set to three and the sub-dimension is computed as

$$\lfloor S/1.5 \rfloor,$$

where S denotes the number of sensors. Moreover, the *dilatation* parameter is the dilatation used in the first dilated convolution, and is computed as

$$\lfloor T/64 \rfloor,$$

where T denotes the length of the time series. On the other hand, another parameter that we have had to tune has been the kernel sizes in InceptionTime. Here we have differentiated the elevator use case and the benchmark datasets because the difference in the length of the series is considerable. On the one hand, in the elevator use case, the length of the series is 8000 points, so we have considered large kernels of size 49, 99, and 199 in this case. On the other hand, in the benchmark datasets, as the series is of 500 points, we have considered kernels of size 9, 19, and 39.

Table 9.4: Hyperparameters selected for benchmark algorithms.

Algorithm	Hyperparameter	Dataset			
		Elevator	**SMD**	**MSL**	**SMAP**
MLSTM-FCN	*Conv 1 (filters, kernel size, stride)*	(128,8,1)	(128,8,1)	(128,8,1)	(128,8,1)
	Conv 2 (filters, kernel size, stride)	(256,5,1)	(256,5,1)	(256,5,1)	(256,5,1)
	Conv 3 (filters, kernel size, stride)	(128,3,1)	(128,3,1)	(128,3,1)	(128,3,1)
	Conv dropout rate	0.3	0.3	0.3	0.3
	Lstm layers	1	1	1	1
	Lstm units	128	128	128	128
	Lstm dropout rate	0.8	0.8	0.8	0.8
	Dim feedforward	128	128	128	128
TapNet	*Conv 1 (filters, kernel size, stride)*	(256,8,1)	(256,8,1)	(256,8,1)	(256,8,1)
	Conv 2 (filters, kernel size, stride)	(256,5,1)	(256,5,1)	(256,5,1)	(256,5,1)
	Conv 3 (filters, kernel size, stride)	(128,3,1)	(128,3,1)	(128,3,1)	(128,3,1)
	Lstm units	128	128	128	128
	Dilatation	125	7	7	7
	Rp params	(3,13)	(3,25)	(3,36)	(3,16)
InceptionTime	*Inception blocks*	3	3	3	3
	Block 1 (in channels, filters, kernel sizes, bottleneck channels, use residual, activation)	(20,32,[49,99,199],32,True,ReLU)	(38,32,[9,19,39],32,True,ReLU)	(55,32,[9,19,39],32,True,ReLU)	(25,32,[9,19,39],32,True,ReLU)
	Block 2 (in channels, filters, kernel sizes, bottleneck channels, use residual, activation)	(128,32,[49,99,199],32,True,ReLU)	(128,32,[9,19,39],32,True,ReLU)	(128,32,[9,19,39],32,True,ReLU)	(128,32,[9,19,39],32,True,ReLU)
	Block 3 (in channels, filters, kernel sizes, bottleneck channels, use residual, activation)	(128,32,[49,99,199],32,True,ReLU)	(128,32,[9,19,199],32,True.ReLU)	(128,32,[9,19,39],32,True,ReLU)	(128,32,[9,19,39],32,True,ReLU)
	Maxpool (kernel size, stride)	(3,1)	(3,1)	(3,1)	(3,1)
MH 1DCNN	*Convolutional blocks*	3	3	3	3
	Conv (filters, kernel size, stride)	(20,5,1)	(20,5,1)	(20,5,1)	(20,5,1)
	Maxpool (kernel size, stride)	(2,2)	(2,2)	(2.2)	(2,2)
	Dropout rate	0.1	0.1	0.1	0.1
	Lstm units	128	128	128º	128

9.6 Discussion on number of parameters and training time

9.6.1 Number of parameters and training time in the conducted experiments

Table 9.5 shows each algorithm's learnable parameters and the time it takes to train for each epoch. As we can see, in the elevator use-case, MH1DCNN-LSTM is the model that has the most parameters, followed by DFStrans, which is the model with the most parameters in the other use cases. Generally, training DFSTrans takes longer than the other algorithms except in the elevator use-case, where MLSTM-FCN is the slowest algorithm.t. The algorithm that has fewer parameters to train is MLSTM-FCN. The fastest algorithms are TapNet and InceptionTime.

Table 9.5: Training time and number of parameters.

Algorithm	Dataset	Number of parameters	Training time (s/epochs)
DFStrans	Elevator	3878033	102.14
	SMD	5840153	3.36
	SMAP	2225553	19.24
	MSL	4197153	8.11
InceptionTime	Elevator	3303681	131.29
	SMD	726657	2.10
	SMAP	724161	10.48
	MSL	729921	2.54
TapNet	Elevator	5155962	96.83
	SMD	1389690	2.62
	SMAP	1334394	8.23
	MSL	1457274	2.32
MLSTM-FCN	Elevator	371457	203.21
	SMD	399105	2.49
	SMAP	379137	9.81
	MSL	425217	2.66
MH1DCNN-LSTM	Elevator	20764385	73.48
	SMD	2696145	3.51
	SMAP	1841785	25.32
	MSL	2768905	7.24

9.6.2 Effect of data size on the number of parameters

In this section, we study how the model parameters grow as the spatio-temporal dimensions of the multi-sensor systems grow. As the learnable parameters of the model do not depend on the number of time segments, we

have studied how the model's number of parameters changes with a different number of sensors. Table 9.6 shows the number of DFStrans parameters for different numbers sensors (being $N_w = 20$). These results have been plotted in Figure 9.3, where it can be seen that the number of model parameters grows linearly with the number of sensors.

Number of sensors, $N_w = 20$	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Number of parameters	2606433	3878033	5149633	6421233	7692833	8964433	10236033	11507633

Table 9.6: Number of model parameters for different number of sensors.

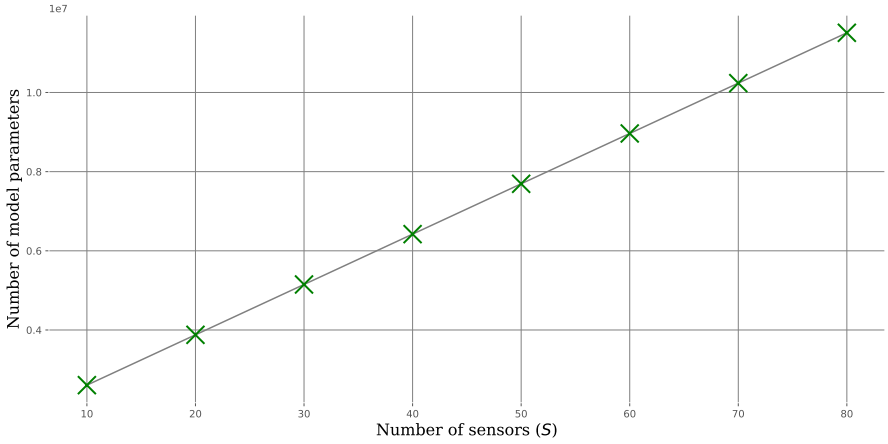


Fig. 9.3: This figure shows the linear relationship between model parameters and S.

References

1. U. Kamath and J. Liu, *Explainable artificial intelligence: An introduction to interpretable machine learning*. Springer, 2021.
2. J. W. Tukey, *Exploratory data analysis*, vol. 2. Reading, Mass., 1977.
3. H. Chen, U. Soni, Y. Lu, R. Maciejewski, and S. Kobourov, "Same stats, different graphs," in *International Symposium on Graph Drawing and Network Visualization*, pp. 463–477, Springer, 2018.
4. J. Matejka and G. Fitzmaurice, "Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1290–1294, 2017.
5. I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.
6. J. V. Stone, "Independent component analysis: an introduction," *Trends in cognitive sciences*, vol. 6, no. 2, pp. 59–64, 2002.
7. L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
8. L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.
9. J. Soler, F. Tencé, L. Gaubert, and C. Buche, "Data clustering and similarity," in *The Twenty-Sixth International FLAIRS Conference*, Citeseer, 2013.
10. L. K. P. J. RDUSSEEUN and P. KAUFMAN, "Clustering by means of medoids," 1987.
11. B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Advances in neural information processing systems*, pp. 2280–2288, 2016.
12. R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
13. B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Machine Learning*, vol. 102, no. 3, pp. 349–391, 2016.
14. "Ann-dt: an algorithm for extraction of decision trees from artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1392–1401, 1999.

15. J. Jung, C. Concannon, R. Shroff, S. Goel, and D. G. Goldstein, "Simple rules for complex decisions," *Available at SSRN 2919024*, 2017.
16. H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1675–1684, 2016.
17. R. Caruana, H. Kangarloo, J. D. Dionisio, U. Sinha, and D. Johnson, "Case-based explanation of non-case-based learning methods.," *Proceedings / AMIA. Annual Symposium. AMIA Symposium*, pp. 212–215, 1999.
18. Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. Part F128815, pp. 623–631, 2013.
19. F. Wang and C. Rudin, "Falling rule lists," in *Artificial Intelligence and Statistics*, pp. 1013–1022, 2015.
20. D. Wei, S. Dash, T. Gao, and O. Gunluk, "Generalized linear rule models," in *International conference on machine learning*, pp. 6687–6696, PMLR, 2019.
21. Z. C. Lipton, "The Mythos of Model Interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
22. C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistic Surveys*, vol. 16, pp. 1–85, 2022.
23. S. Saisubramanian, S. Galhotra, and S. Zilberstein, "Balancing the tradeoff between clustering value and interpretability," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 351–357, 2020.
24. T. Mori and N. Uchihira, "Balancing the trade-off between accuracy and interpretability in software defect prediction," *Empirical Software Engineering*, vol. 24, pp. 779–825, 2019.
25. E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems*, pp. 3504–3512, 2016.
26. J. Heo, H. B. Lee, S. Kim, J. Lee, K. J. Kim, E. Yang, and S. J. Hwang, "Uncertainty-aware attention for reliable interpretation and prediction," in *Advances in Neural Information Processing Systems*, pp. 909–918, 2018.
27. T. Bai, S. Zhang, B. L. Egleston, and S. Vucetic, "Interpretable representation learning for healthcare via capturing disease progression through time," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 43–51, 2018.
28. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.
29. M. Du, F. Li, G. Zheng, and V. 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*, pp. 1285–1298, 2017.
30. A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

31. A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pp. 1–8, 2018.
32. I. Giurgiu and A. Schumann, "Explainable failure predictions with rnn classifiers based on time series data," *arXiv preprint arXiv:1901.08554*, 2019.
33. C. Schockaert, R. Leperlier, and A. Moawad, "Attention mechanism for multivariate time series recurrent model interpretability applied to the ironmaking industry,"
34. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proc. of the IEEE international conference on computer vision*, pp. 618–626, 2017.
35. R. Assaf and A. Schumann, "Explainable deep neural networks for multivariate time series predictions.," in *IJCAI*, pp. 6488–6490, 2019.
36. C. Wang, T. Onishi, K. Nemoto, and K.-L. Ma, "Visual reasoning of feature attribution with deep recurrent neural networks," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1661–1668, IEEE, 2018.
37. C. Wang, X. Wang, and K.-L. Ma, "Visual summary of value-level feature attribution in prediction classes with recurrent neural networks," *arXiv preprint arXiv:2001.08379*, 2020.
38. F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2017.
39. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.
40. Y. Qin, K. Kamnitsas, S. Ancha, J. Nanavati, G. Cottrell, A. Criminisi, and A. Nori, "Autofocus layer for semantic segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 603–611, Springer, 2018.
41. B. Vasu and C. Long, "Iterative and adaptive sampling with spatial attention for black-box model explanations," in *The IEEE Winter Conference on Applications of Computer Vision*, pp. 2960–2969, 2020.
42. V. Shitole, F. Li, M. Kahng, P. Tadepalli, and A. Fern, "One explanation is not enough: structured attention graphs for image classification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11352–11363, 2021.
43. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.
44. B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.
45. Q. Zhang, X. Wang, Y. N. Wu, H. Zhou, and S.-C. Zhu, "Interpretable cnns for object classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
46. W. Yu, K. Yang, Y. Bai, T. Xiao, H. Yao, and Y. Rui, "Visualizing and comparing alexnet and vgg using deconvolutional layers," in *Proceedings of the 33 rd International Conference on Machine Learning*, 2016.

47. S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, and S. Ahmed, "Tsviz: Demystification of deep learning models for time-series analysis," *IEEE Access*, vol. 7, pp. 67027–67040, 2019.
48. J. Pereira and M. Silveira, "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 1275–1282, IEEE, 2018.
49. K. Xu, D. H. Park, C. Yi, and C. Sutton, "Interpreting deep classifier by visual distillation of dark knowledge," *arXiv preprint arXiv:1803.04042*, 2018.
50. M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "A unified view of gradient-based attribution methods for deep neural networks," in *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning-Now What?(NIPS 2017)*, ETH Zurich, 2017.
51. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: visualising image classification models and saliency maps," in *Proceedings of the International Conference on Learning Representations (ICLR)*, ICLR, 2014.
52. A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.
53. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.
54. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, pp. 3145–3153, PMLR, 2017.
55. M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328, JMLR. org, 2017.
56. R. Luss, P.-Y. Chen, A. Dhurandhar, P. Sattigeri, K. Shanmugam, and C.-C. Tu, "Generating contrastive explanations with monotonic attribute functions," *arXiv preprint arXiv:1905.12698*, 2019.
57. M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.
58. S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis.," in *ISMIR*, pp. 537–543, 2017.
59. T. Peltola, "Local interpretable model-agnostic explanations of bayesian predictive models via kullback-leibler projections," in *Workshop on Explainable Artificial Intelligence*, 2018.
60. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.
61. R. Rodríguez-Pérez and J. Bajorath, "Interpretation of compound activity predictions from complex machine learning models using local approximations and shapley values," *Journal of medicinal chemistry*, 2019.
62. S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations

to global understanding with explainable ai for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 2522–5839, 2020.
63. D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.
64. S. Tan, M. Soloviev, G. Hooker, and M. Wells, "Tree space prototypes: Another look at making tree ensembles interpretable," *Foundations of data science*, 2020.
65. R. Khanna, B. Kim, J. Ghosh, and S. Koyejo, "Interpreting black box predictions using fisher kernels," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3382–3390, PMLR, 2019.
66. S. Fort, "Gaussian prototypical networks for few-shot learning on omniglot," *arXiv preprint arXiv:1708.02735*, 2017.
67. O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
68. A. H. Gee, D. Garcia-Olano, J. Ghosh, and D. Paydarfar, "Explaining deep classification of time-series data with learned prototypes," in *CEUR workshop proceedings*, vol. 2429, p. 15, NIH Public Access, 2019.
69. S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.
70. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
71. A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Advances in Neural Information Processing Systems*, pp. 592–603, 2018.
72. A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
73. A. Van Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," *arXiv preprint arXiv:1907.02584*, 2019.
74. S. Rathi, "Generating counterfactual and contrastive explanations using shap," *arXiv preprint arXiv:1906.09293*, 2019.
75. J. H. Friedman, B. E. Popescu, *et al.*, "Predictive learning via rule ensembles," *The Annals of Applied Statistics*, vol. 2, no. 3, pp. 916–954, 2008.
76. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
77. M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
78. J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European conference on machine learning*, pp. 437–448, Springer, 2005.
79. R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," *arXiv preprint arXiv:1805.10820*, 2018.
80. G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

81. H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
82. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
83. H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328, IEEE, 2008.
84. B. Das, N. C. Krishnan, and D. J. Cook, "Racog and wracog: Two probabilistic oversampling techniques," *IEEE transactions on knowledge and data engineering*, vol. 27, no. 1, pp. 222–234, 2014.
85. J. Prusa, T. M. Khoshgoftaar, D. J. Dittman, and A. Napolitano, "Using random undersampling to alleviate class imbalance on tweet sentiment data," in *2015 IEEE international conference on information reuse and integration*, pp. 197–202, IEEE, 2015.
86. I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126, 2003.
87. P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE transactions on information theory*, vol. 14, no. 3, pp. 515–516, 1968.
88. R. K. Dhurjad and S. Banait, "Imbalanced time series data classification using oversampling technique," *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)*, p. 75, 2015.
89. N. Moniz, P. Branco, and L. Torgo, "Resampling strategies for imbalanced time series forecasting," *International Journal of Data Science and Analytics*, vol. 3, no. 3, pp. 161–181, 2017.
90. T. Zhu, Y. Lin, and Y. Liu, "Oversampling for imbalanced time series data," *arXiv preprint arXiv:2004.06373*, 2020.
91. B. Krawczyk, M. Woźniak, and G. Schaefer, "Cost-sensitive decision tree ensembles for effective imbalanced classification," *Applied Soft Computing*, vol. 14, pp. 554–562, 2014.
92. Y. Park and J. Ghosh, "Ensembles of α-trees for imbalanced classification problems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 131–143, 2012.
93. L. Lin, R. Zuo, S. Yang, and Z. Zhang, "Svm ensemble for anomaly detection based on rotation forest," in *2012 Third International Conference on Intelligent Control and Information Processing*, pp. 150–153, IEEE, 2012.
94. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: misclassification cost-sensitive boosting," in *Icml*, vol. 99, pp. 97–105, 1999.
95. N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *European conference on principles of data mining and knowledge discovery*, pp. 107–119, Springer, 2003.
96. C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2009.

97. J. Frery, A. Habrard, M. Sebban, O. Caelen, and L. He-Guelton, "Efficient top rank optimization with gradient boosting for supervised anomaly detection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 20–35, Springer, 2017.
98. H. Mahmoud, W. Wu, and M. M. Gaber, "A time-series self-supervised learning approach to detection of cyber-physical attacks in water distribution systems," *Energies*, vol. 15, no. 3, p. 914, 2022.
99. A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "Water leak detection using self-supervised time series classification," *Information Sciences*, vol. 574, pp. 528–541, 2021.
100. J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hivecote: The hierarchical vote collective of transformation-based ensembles," *ACM transactions on knowledge discovery from data*, vol. 12, no. 5, 2018.
101. Y. Jiao, K. Yang, D. Song, and D. Tao, "Timeautoad: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1604–1619, 2022.
102. J. Ircio, A. Lojo, J. A. Lozano, and U. Mori, "A multivariate time series streaming classifier for predicting hard drive failures [application notes]," *IEEE Computational Intelligence Magazine*, vol. 17, no. 1, pp. 102–114, 2022.
103. J. Ircio, A. Lojo, U. Mori, and J. A. Lozano, "Mutual information based feature subset selection in multivariate time series classification," *Pattern Recognit.*, vol. 108, p. 107525, 2020.
104. J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
105. M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, pp. 197–205, IEEE, 2014.
106. M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head cnn–rnn for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019.
107. T. Wen and R. Keyes, "Time series anomaly detection using convolutional neural networks and transfer learning," *arXiv preprint arXiv:1905.13628*, 2019.
108. Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.
109. G. Marín, P. Casas, and G. Capdehourat, "Rawpower: Deep learning based anomaly detection from raw network traffic measurements," in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pp. 75–77, 2018.
110. M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pp. 72–83, 2018.
111. H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at Microsoft," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 3330680, no. c, pp. 3009–3017, 2019.

112. F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.
113. S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats," *Computers in biology and medicine*, vol. 102, pp. 278–287, 2018.
114. G. Paragliola and A. Coronato, "Gait anomaly detection of subjects with parkinson's disease using a deep time series-based approach," *IEEE Access*, vol. 6, pp. 73280–73292, 2018.
115. T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.
116. X. Xie, B. Wang, T. Wan, and W. Tang, "Multivariate abnormal detection for industrial control systems using 1d cnn and gru," *IEEE Access*, vol. 8, pp. 88348–88359, 2020.
117. F. Liu, X. Zhou, J. Cao, Z. Wang, T. Wang, H. Wang, and Y. Zhang, "Anomaly Detection in Quasi-Periodic Time Series Based on Automatic Data Segmentation and Attentional LSTM-CNN," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2626–2640, 2022.
118. W. Groß, S. Lange, J. Bödecker, and M. Blum, "Predicting time series with space-time convolutional and recurrent neural networks.," in *ESANN*, 2017.
119. Q. Du, W. Gu, L. Zhang, and S.-L. Huang, "Attention-based lstm-cnns for time-series classification," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pp. 410–411, ACM, 2018.
120. D. Hu, "An introductory survey on attention mechanisms in nlp problems," in *Proceedings of SAI Intelligent Systems Conference*, pp. 432–448, Springer, 2019.
121. Y. Using Extreme Learning Machine for Intrusion Detection in a Big Data Environment Junlong and L. Sukhostat, "Anomaly detection in network traffic using extreme learning machine," in *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–4, IEEE, 2016.
122. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.
123. Y. Wang, D. Li, Y. Du, and Z. Pan, "Anomaly detection in traffic using l1-norm minimization extreme learning machine," *Neurocomputing*, vol. 149, pp. 415–425, 2015.
124. J. Xiang, M. Westerlund, D. Sovilj, and G. Pulkkis, "Using extreme learning machine for intrusion detection in a big data environment," in *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, pp. 73–82, 2014.
125. A. Iturria, J. Labaien, S. Charramendieta, A. Lojo, J. Del Ser, and F. Herrera, "A framework for adapting online prediction algorithms to outlier detection over time series," *Knowledge-Based Systems*, vol. 256, p. 109823, 2022.
126. J.-M. Park and J.-H. Kim, "Online recurrent extreme learning machine and its application to time-series prediction," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1983–1990, IEEE, 2017.
127. M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *Ieee Access*, vol. 7, pp. 1991–2005, 2018.

128. N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1939–1947, 2015.
129. Y. He and J. Zhao, "Temporal Convolutional Networks for Anomaly Detection in Time Series," *Journal of Physics: Conference Series*, vol. 1213, no. 4, 2019.
130. V. L. Cao and J. Mcdermott, "Sinistralité au travail:Des tendances d'évolution différenciées selon le sexe," *Agence nationale pour l'amélioration des conditions de travail*.
131. L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.
132. S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015*, no. October, 2015.
133. G. Qin, Y. Chen, and Y.-X. Lin, "Anomaly detection using lstm in ip networks," in *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 334–337, IEEE, 2018.
134. K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 387–395, 2018.
135. J. S. Hunter, "The exponentially weighted moving average," *Journal of quality technology*, vol. 18, no. 4, pp. 203–210, 1986.
136. J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, no. January, pp. 140–145, 2017.
137. N. Ding, H. X. Ma, H. Gao, Y. H. Ma, and G. Z. Tan, "Real-time anomaly detection based on long short-Term memory and Gaussian Mixture Model," *Computers and Electrical Engineering*, vol. 79, 2019.
138. Z. Zhong, A. Y. Sun, Q. Yang, and Q. Ouyang, "A deep learning approach to anomaly detection in geological carbon sequestration sites using pressure measurements," *Journal of hydrology*, vol. 573, pp. 885–894, 2019.
139. A. Deng and B. Hooi, "Graph Neural Network-Based Anomaly Detection in Multivariate Time Series," *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, vol. 5A, pp. 4027–4035, 2021.
140. Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9179–9189, 2022.
141. M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," *ACM International Conference Proceeding Series*, vol. 02-Decembe, pp. 4–11, 2014.
142. P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," tech. rep.
143. B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–19, 2018.

144. C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 1409–1416, 2019.
145. J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised Anomaly Detection on Multivariate Time Series," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3395–3404, 2020.
146. D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.
147. H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. BT - Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018," vol. 2, pp. 187–196, 2018.
148. W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised Anomaly Detection for Intricate KPIs via Adversarial Training of VAE," *Proceedings - IEEE INFOCOM*, vol. 2019-April, pp. 1891–1899, 2019.
149. Y. Su, R. Liu, Y. Zhao, W. Sun, C. Niu, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 1485, pp. 2828–2837, 2019.
150. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
151. D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.
152. G. Kitagawa and W. Gersch, "Linear gaussian state space modeling," in *Smoothness priors analysis of time series*, pp. 55–65, Springer, 1996.
153. H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. 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*, pp. 187–196, 2018.
154. A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1067–1075, 2017.
155. B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "Beatgan: Anomalous rhythm detection using adversarially generated time series," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-Augus, pp. 4433–4439, 2019.
156. X. Chen, L. Deng, F. Huang, C. Zhang, Z. Zhang, Y. Zhao, and K. Zheng, "DAEMON: Unsupervised anomaly detection and interpretation for multivariate time series," *Proceedings - International Conference on Data Engineering*, vol. 2021-April, pp. 2225–2230, 2021.
157. Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.
158. J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy," 2021.

159. Y. Zhang, Y. Chen, J. Wang, and Z. Pan, "Unsupervised deep anomaly detection for multi-sensor time-series signals," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
160. X. Wang, D. Pi, X. Zhang, H. Liu, and C. Guo, "Variational transformer-based anomaly detection approach for multivariate time series," *Measurement*, vol. 191, p. 110791, 2022.
161. S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data," 2022.
162. L. Yu, "DTAAD: Dual Tcn-Attention Networks for Anomaly Detection in Multivariate Time Series Data," pp. 1–14, 2023.
163. T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3127–3141, 2020.
164. L. Shen, Z. Li, and J. T. Kwok, "Thoc," *Advances in Neural Information Processing Systems*, vol. 2020-Decem, no. NeurIPS, pp. 1–11, 2020.
165. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.
166. Y. Qin and Y. Lou, "Hydrological time series anomaly pattern detection based on isolation forest," *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, no. Itnec, pp. 1706–1710, 2019.
167. P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, "K-Means-based isolation forest," *Knowledge-Based Systems*, vol. 195, p. 105659, 2020.
168. M. Chater, A. Borgi, M. T. Slama, K. Sfar-Gandoura, and M. I. Landoulsi, "Fuzzy Isolation Forest for Anomaly Detection," *Procedia Computer Science*, vol. 207, pp. 916–925, 2022.
169. C. Y. Priyanto, H. D. Purnomo, *et al.*, "Combination of isolation forest and lstm autoencoder for anomaly detection," in *2021 2nd International Conference on Innovative and Creative Information Technology (ICITech)*, pp. 35–38, IEEE, 2021.
170. P. H. Tran, C. Heuchenne, and S. Thomassey, "An anomaly detection approach based on the combination of lstm autoencoder and isolation forest for multivariate time series data," in *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, pp. 589–596, World Scientific, 2020.
171. Y. Ma, Q. Zhang, J. Ding, Q. Wang, and J. Ma, "Short term load forecasting based on iforest-lstm," in *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2278–2282, IEEE, 2019.
172. M. Çelik, F. Dadaşer-Çelik, and A. Dokuz, "Anomaly detection in temperature data using DBSCAN algorithm," *INISTA 2011 - 2011 International Symposium on INnovations in Intelligent SysTems and Applications*, pp. 91–95, 2011.
173. P. Tan, M. Steinbach, V. Kumar, C. Potter, S. Klooster, and A. Torregrosa, "Finding spatio-temporal patterns in earth science data," in *KDD 2001 Workshop on Temporal Data Mining*, vol. 19, 2001.
174. S. Wibisono, M. T. Anwar, A. Supriyanto, and I. H. Amin, "Multivariate weather anomaly detection using DBSCAN clustering algorithm," *Journal of Physics: Conference Series*, vol. 1869, no. 1, 2021.
175. P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers-a tutorial," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

176. S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.
177. F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*, pp. 15–27, Springer, 2002.
178. M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.
179. W. Jin, A. K. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 577–593, Springer, 2006.
180. F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 811–820, 2007.
181. F. Angiulli and F. Fassetti, "Distance-based outlier queries in data streams: the novel task and algorithms," *Data Mining and Knowledge Discovery*, vol. 20, no. 2, pp. 290–324, 2010.
182. V. Ishimtsev, A. Bernstein, E. Burnaev, and I. Nazarov, "Conformal k-nn anomaly detector for univariate data streams," in *Conformal and Probabilistic Prediction and Applications*, pp. 213–227, 2017.
183. Y. Ikeda, K. Tajiri, Y. Nakano, K. Watanabe, and K. Ishibashi, "Estimation of dimensions contributing to detected anomalies with variational autoencoders," *arXiv preprint arXiv:1811.04576*, 2018.
184. G.-b. Jang and S.-B. Cho, "Anomaly detection of 2.4 l diesel engine using one-class svm with variational autoencoder," in *Annual Conference of the PHM Society*, vol. 11, 2019.
185. Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. 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*, pp. 2828–2837, 2019.
186. K. Zhou, S. Gao, J. Cheng, Z. Gu, H. Fu, Z. Tu, J. Yang, Y. Zhao, and J. Liu, "Sparse-gan: Sparsity-constrained generative adversarial network for anomaly detection in retinal oct image," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 1227–1231, IEEE, 2020.
187. L. Antwarg, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using shap," *arXiv preprint arXiv:1903.02407*, 2019.
188. N. Takeishi, "Shapley values of reconstruction errors of pca for explaining anomaly detection," in *2019 International Conference on Data Mining Workshops (ICDMW)*, pp. 793–798, IEEE, 2019.
189. I. Giurgiu and A. Schumann, "Additive explanations for anomalies detected from multivariate temporal data," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2245–2248, 2019.
190. J.-R. Rehse, N. Mehdiyev, and P. Fettke, "Towards explainable process predictions for industry 4.0 in the dfki-smart-lego-factory," *KI-Künstliche Intelligenz*, vol. 33, no. 2, pp. 181–187, 2019.
191. J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decision Support Systems*, vol. 100, pp. 129–140, 2017.

192. M. Carletti, C. Masiero, A. Beghi, and G. A. Susto, "Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 21–26, IEEE, 2019.
193. A. Barbado, Ó. Corcho, and R. Benjamins, "Rule extraction in unsupervised anomaly detection for model explainability: Application to oneclass svm," *Expert Systems with Applications*, vol. 189, p. 116100, 2022.
194. D. Martens, J. Huysmans, R. Setiono, J. Vanthienen, and B. Baesens, "Rule extraction from support vector machines: an overview of issues and application in credit scoring," in *Rule extraction from support vector machines*, pp. 33–63, Springer, 2008.
195. M. Kopp, T. Pevnỳ, and M. Holeňa, "Anomaly explanation with random forests," *Expert Systems with Applications*, p. 113187, 2020.
196. V. T. Trifunov, M. Shadaydeh, B. Barz, and J. Denzler, "Anomaly attribution of multivariate time series using counterfactual reasoning," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 166–172, IEEE, 2021.
197. E. Delaney, D. Greene, and M. T. Keane, "Instance-based counterfactual explanations for time series classification," in *Case-Based Reasoning Research and Development: 29th International Conference, ICCBR 2021, Salamanca, Spain, September 13–16, 2021, Proceedings 29*, pp. 32–47, Springer, 2021.
198. H. Cheng, D. Xu, S. Yuan, and X. Wu, "Fine-grained anomaly detection in sequential data via counterfactual explanations," *arXiv preprint arXiv:2210.04145*, 2022.
199. A. Van Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pp. 650–665, Springer, 2021.
200. T. D. Duong, Q. Li, and G. Xu, "Prototype-based counterfactual explanation for causal classification," *arXiv preprint arXiv:2105.00703*, 2021.
201. S. Filali Boubrahimi and S. M. Hamdi, "On the mining of time series data counterfactual explanations using barycenters," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3943–3947, 2022.
202. R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617, 2020.
203. D. Sulem, M. Donini, M. B. Zafar, F.-X. Aubet, J. Gasthaus, T. Januschowski, S. Das, K. Kenthapadi, and C. Archambeau, "Diverse counterfactual explanations for anomaly detection in time series," *arXiv preprint arXiv:2203.11103*, 2022.
204. J. Crabbé and M. Van Der Schaar, "Explaining time series predictions with dynamic masks," in *International Conference on Machine Learning*, pp. 2166–2177, PMLR, 2021.
205. M. Schemmer, J. Holstein, N. Bauer, N. Kühl, and G. Satzger, "Towards meaningful anomaly detection: The effect of counterfactual explanations on the investigation of anomalies in multivariate time series," *arXiv preprint arXiv:2302.03302*, 2023.

206. W. Todo, M. Selmani, B. Laurent, and J.-M. Loubes, "Counterfactual explanation for multivariate times series using a contrastive variational autoencoder," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.
207. J. Labaien Soto, E. Zugasti Uriguen, and X. De Carlos Garcia, "Real-time, model-agnostic and user-driven counterfactual explanations using autoencoders," *Applied Sciences*, vol. 13, no. 5, p. 2912, 2023.
208. C. Xiao, X. Xu, Y. Lei, K. Zhang, S. Liu, and F. Zhou, "Counterfactual graph learning for anomaly detection on attributed networks," *IEEE Transactions on Knowledge and Data Engineering*, 2023.
209. M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.
210. C. Ding, S. Sun, and J. Zhao, "Mst-gat: A multimodal spatial–temporal graph attention network for time series anomaly detection," *Information Fusion*, vol. 89, pp. 527–536, 2023.
211. K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.
212. S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data," *Proceedings of VLDB*, vol. 15, no. 6, pp. 1201–1214, 2022.
213. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
214. D. H. Ballard, "Modular learning in neural networks.," in *AAAI*, pp. 279–284, 1987.
215. J. Zhao, F. Deng, Y. Cai, and J. Chen, "Long short-term memory-fully connected (lstm-fc) neural network for pm2. 5 concentration prediction," *Chemosphere*, vol. 220, pp. 486–492, 2019.
216. S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," *arXiv preprint arXiv:2010.10596*, 2020.
217. E. M. Kenny and M. T. Keane, "On generating plausible counterfactual and semi-factual explanations for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11575–11585, 2021.
218. D. Nemirovsky, N. Thiebaut, Y. Xu, and A. Gupta, "Countergan: Generating realistic counterfactuals with residual generative adversarial nets," *arXiv preprint arXiv:2009.05199*, 2020.
219. S. Liu, B. Kailkhura, D. Loveland, and Y. Han, "Generative counterfactual introspection for explainable deep learning," in *2019 IEEE global conference on signal and information processing (GlobalSIP)*, pp. 1–5, IEEE, 2019.
220. D. Mahajan, C. Tan, and A. Sharma, "Preserving causal constraints in counterfactual explanations for machine learning classifiers," *arXiv preprint arXiv:1912.03277*, 2019.
221. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
222. "Mathworks gearbox simulator." https://www.mathworks.com/help/signal/examples/vibration-analysis-of-rotating-machinery.html.

223. M. Lin, H. C. Lucas Jr, and G. Shmueli, "Research commentary—too big to fail: large samples and the p-value problem," *Information Systems Research*, vol. 24, no. 4, pp. 906–917, 2013.
224. F. Hvilshøj, A. Iosifidis, and I. Assent, "On quantitative evaluations of counterfactuals," *arXiv preprint arXiv:2111.00177*, 2021.
225. W. Li, W. Hu, N. Chen, and C. Feng, "Stacking VAE with graph neural networks for effective and interpretable time series anomaly detection," *arXiv preprint arXiv:2105.08397*, 2021.
226. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.
227. E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges, "A spatio-temporal transformer for 3d human motion prediction," in *2021 International Conference on 3D Vision (3DV)*, pp. 565–574, IEEE, 2021.
228. M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, "Gated transformer networks for multivariate time series classification," *arXiv preprint arXiv:2103.14438*, 2021.
229. B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning spatio-temporal transformer for visual tracking," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10448–10457, 2021.
230. L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting," *Transactions in GIS*, vol. 24, no. 3, pp. 736–755, 2020.
231. C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-temporal graph transformer networks for pedestrian trajectory prediction," in *European Conference on Computer Vision*, pp. 507–523, Springer, 2020.
232. J. Grigsby, Z. Wang, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint arXiv:2109.12218*, 2021.
233. E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges, "A spatio-temporal transformer for 3d human motion prediction," in *2021 International Conference on 3D Vision (3DV)*, pp. 565–574, IEEE, 2021.
234. M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019.
235. M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
236. C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
237. X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 6845–6852, 2020.
238. H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding AlexNet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.
239. F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.
240. K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresh-

olding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.
241. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
242. J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *International Conference on Learning Representations*, 2021.
243. S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.
244. J. S. Labaien, T. Idé, P.-Y. Chen, X. D. Carlos, and E. Zugasti, "Diagnostic spatio-temporal transformer with faithful encoding," 2023. To be submitted.
245. H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 841–850, IEEE, 2020.
246. D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International conference on artificial neural networks*, pp. 703–716, Springer, 2019.
247. A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 38–44, IEEE, 2015.
248. R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
249. A. P. Mathur and N. O. 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)*, pp. 31–36, IEEE, 2016.
250. R. Mark, G. Moody, and S. Greenwald, "Mit-bih supraventricular arrhythmia database," 1990.
251. S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 7194–7201, 2022.
252. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.
253. K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.