University of Nottingham
UK | CHINA | MALAYSIA

Advancing Explainability in Multivariate Time Series Classification through Generative Models

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy, May 2024.**

Han Meng

20245258

Supervised by

Isaac Triguero
Christian Wagner

Signature Han MENG

Date 20 / 05 / 2024

Abstract

As Artificial Intelligence (AI) techniques become increasingly integrated in various domains, concerns about their interpretability have increased. This has catalysed the development of eXplainable AI (XAI), which seeks to make AI systems more understandable to humans. However, most XAI research has focused on tabular data, images, or language models, with time series models receiving limited attention despite their pervasiveness. This thesis aims to advance XAI specifically within the context of Multivariate Time Series Classification (MTSC) problems, providing varied explanation types to meet diverse user needs.

Among the various types of explanations, feature importance stands out as one of the most intuitive, where the most relevant features influencing a model's decisions are identified. Most of the existing feature importance explanation techniques generate samples to feed into the model with the aim of probing the model's mechanisms. However, the methods used for generating samples often disrupt the inherent temporal dependencies within time series data. This disruption leads to samples that are misaligned with the distribution of the training data, resulting in misleading explanations. This thesis introduces a model-agnostic framework that incorporates a generative model trained to estimate the training data distribution, thereby generating within-distribution samples for accurate feature importance evaluation. Furthermore, time series models typically incorporate a large number of input features, challenging the identification of the most relevant ones. To address this difficulty, the framework integrates an optimisation method to efficiently identify the most important features.

In addition to the challenges in accurately identifying the most important features, current research also highlights the lack of stability as a key issue affecting most feature importance explanation methods, such as the well-known Local Interpretation Model-agnostic Explanation (LIME). This means that the explanations they provide for the same instance are often inconsistent. Although various methods have been developed to mitigate this instability, the underlying processes driving the instability in the context of MTSC remain somewhat obscure. This thesis explores this area and

highlights that a significant but often overlooked issue in current research affecting stability is also the out-of-distribution samples generated during the explanation process. To address this, a novel framework is proposed which employs a generative model and a local sampling technique to produce within-distribution neighbouring data for the instance to be explained. Additionally, it integrates an adaptive weighting strategy to facilitate hyperparameter optimisation, thereby further enhancing the stability of the explanations.

To advance XAI in the field of MTSC, relying solely on feature importance may not suffice, for example for the general public, who usually seek more straightforward explanations to assist their decision-making processes. In this context, counterfactual explanations are promising. However, current counterfactual methods for MTSC usually fail to provide plausible and meaningful explanations, often overlooking the distribution of training data during the generation of explanations. This thesis proposes two novel approaches to improve the viability of counterfactual explanations for MTSC. The first method estimates the distribution density of the training samples using Gaussian Mixture Models (GMMs) and then trains the generative model to create counterfactual samples in the densest regions. The second approach adopts a generative adversarial network paradigm, where the generator is trained to create counterfactuals and the discriminator is optimised to distinguish real and fake samples. This approach avoids making specific assumptions about the data distribution, such as the GMMs in the first approaches. Both approaches aim to create counterfactuals using generative models, with the aim of aligning created counterfactuals with the distribution of realistic time series. Experiments carried out with real-world data sets highlight the strengths of each approach in different real-world problems.

In summary, this thesis advances XAI in MTSC by addressing key challenges in feature importance and counterfactual explanations. Throughout this thesis, a strong emphasis is placed on leveraging generative models to produce within-distribution samples with the aim of avoiding the out-of-distribution problem and enhancing explanation reliability. This thesis underpins the promising role of advanced generative models in XAI, setting the stage for their expanded contribution and development in future XAI research.

Acknowledgements

First of all, I extend my deepest gratitude to my supervisors, Dr. Isaac Triguero and Dr. Christian Wagner, for their invaluable advice, unwavering support, and patience throughout my Ph.D. studies. Their extensive knowledge and rich experience have not only guided me through my academic research but also provided immense encouragement in my daily life. I would also like to extend my heartfelt thanks to my colleagues at the Computational Optimisation and Learning (COL) and the Lab for Uncertainty in Data and Decision Making (LUCID) groups. Their collaborative spirit, insightful feedback, and stimulating discussions have been instrumental in shaping my research and professional growth. Last but not least, I cannot express enough thanks to my family and friends for their unwavering support and endless love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, the development of hardware and increased data availability has significantly advanced modern Artificial Intelligence (AI) techniques, such as deep learning (LeCun et al., 2015). These models have achieved promising performance in various fields (Hamet and Tremblay, 2017; Tadapaneni, 2019; Nti et al., 2022). Recently, a notable example in this field is the Chat Generative Pre-trained Transformer (ChatGPT) developed by OpenAI (OpenAI, 2023). ChatGPT has demonstrated remarkable capabilities in a variety of tasks, such as language understanding, code debugging, and story writing, among others. The innovations accompanying it highlight the potential of AI to revolutionise everyday interactions and enhance human productivity, marking the beginning of a new era in AI for humans.

As AI techniques become increasingly integrated into various fields, some concerns about them have also risen. A notable concern is its interpretability. Current modern AI techniques are usually based on deep neural networks, which are often considered black-boxes. These models process input through a large number of parameters to produce final predictions. In this circumstance, users do not often know whether the predictions made by

these models are reliable or unbiased. This lack of interpretability significantly impedes their employment in some safety-critical areas such as finance and medicine. In these areas, the ability to interpret and explain AI decisions is not just desirable but necessary. This urgent need has spurred the development of XAI, which seeks to bridge the gap between AI performance and human understanding (Barredo Arrieta et al., 2020).

The main objective of XAI is to provide explanations to humans to help them understand the reasons behind the predictions. In the XAI literature, explainability refers to two distinct high-level concepts: post-hoc explainability or interpretability, and transparency (also referred to as self-explainability, e.g. in (Ding et al., 2022)). The former captures the ability to explain a given output generated by an AI model, while the latter describes the property of an AI model as being interpretable by a human[1], i.e., including its components, mechanisms, and training process. Regarding the former, post-hoc methods can generally be applied to a wide range of algorithms since many post-hoc methods are model-agnostic (Kamath and Liu, 2021). Moreover, post-hoc methods will not affect the predictive performance of a classifier, while some self-explainable classifiers sacrifice their performance for interpretability (Lipton, 2018). Although it is contested whether self-explainable methods have to sacrifice their performance for interpretability (Rudin, 2019), post-hoc analysis approaches may relieve model builders of the burden of thinking about how to provide explanations and allow them to focus only on predictive performance. In addition, post-hoc analysis can provide different kinds of explanation, such as sample-based counterfactual explanations (R. Fernández et al., 2022) or feature importance-based explanations (Ribeiro et al., 2016; Lundberg and Lee, 2017). However, post-hoc analyses also face the risk of generat-

[1]The notion of who this human is, is a crucial aspect of assessing a model's transparency, i.e. it may be transparent to its creating engineer, but not to an end user.

ing unreliable explanations, which means that the explanations provided by post-hoc approaches may not faithfully reflect how the classifier works (Rudin, 2019).

Explanations in the XAI domain are typically categorised according to their scope and applicability, leading to two primary classifications: global explanations and local explanations (Ding et al., 2022). Global explanations aim to reveal the global behaviour of the classifier on the whole data set, while local explanations focus on the behaviour on a specific instance. Due to the heterogeneity of real-world data sets, a model might behave differently when making predictions on different sets of instances. Thus, the development of local instance-wise explanations has drawn increasing attention in recent years (Le et al., 2023).

XAI has been a growing field of research for many years. However, the majority of XAI efforts have focused on tabular data, images, or language models. Time series models, despite their importance, have not received sufficient attention in XAI research (Barredo Arrieta et al., 2020; Ding et al., 2022). This gap is critical considering the wide application of time series data in various domains such as finance and healthcare. Therefore, the main motivation of this thesis is to advance the development of XAI in the context of time series models.

This chapter is organised as follows. Section 1.1 delves into MTSC, the primary focus of this thesis, and discusses the challenges faced by XAI in this context. Section 1.2 highlights the need for various explanations to satisfy diverse user requirements. Section 1.3 identifies the gaps of current explanation methods when applied in the context of MTSC and outlines the objectives of this thesis. Section 1.4 summarises the main contributions of this thesis. Finally, Section 1.5 provides the overall structure of the thesis.

1.1 Multivariate Time Series Classification within the Context of XAI

A time series is defined as a sequence of real values recorded in chronological order. These series can be univariate, involving data from a single source, or multivariate, where data are gathered from multiple sources. For example, Figure 1.1 illustrates multivariate time series data that represent the operational status of a machine, as captured by four different sensors. In recent years, with rapid technological evolution and the growing accessibility of data, especially through advancements in the internet of things, there has been a significant increase in the availability of such time series data. This abundance of data has catalysed advances in time series related models, such as forecasting and classification models.

Time series forecasting has become pervasive in research, focusing on the prediction of future values by analysing past and present data (Masini et al., 2023). This approach is crucial in various fields, from finance (Sezer et al., 2020) to weather prediction (Karevan and Suykens, 2020), where understanding future trends or conditions based on historical data is essential. In contrast, time series classification diverges from forecasting by concentrating on categorising a time series to discern patterns or events. An example of a time series classification problem is shown in Figure 1.1. This model is built to predict potential machine failures in the future based on the input multivariate time series data. It has various types of applications, such as pattern recognition in financial markets (Doroudyan and Niaki, 2021), health monitoring through wearable devices (Rosafalco et al., 2020), and predictive maintenance in manufacturing industries (Serradilla et al., 2022).

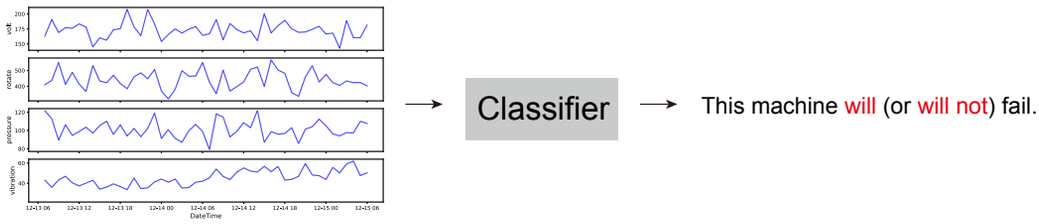


Figure 1.1: A binary MTSC problem. The classifier is built to predict whether the machine will fail later based on the data recorded in the past 48 hours.

In this thesis, the focus will be on classification problems, with a twofold rationale underlying this decision. First, the techniques developed for classification can be adapted to forecasting challenges by reinterpreting forecasting tasks as classification problems. For example, instead of forecasting a specific value, results can be categorised into classes such as 'high', 'medium', or 'low', transforming the forecasting challenge into a classification one. This categorisation also allows the application of interpretability techniques originally designed for classification models to be used in forecast contexts (Assaf et al., 2019). Secondly, the discrete output nature of classification models facilitates the setting of specific objectives for developing explanation methods, which is crucial to improving model interpretability. For example, in a scenario where the classifier predicts a possible machine failure (as depicted in Figure 1.1), the explanation method might focus on answering why the model predicts such an outcome and under which conditions the model will not predict such a failure. This clear and targeted question aids in guiding the design of effective explanation methods. However, when dealing with forecasting models that predict specific values, such as a model that forecasts the price of one stock to be 15 dollars, formulating an effective explanation question is not so straightforward. Questions like 'Why did the model predict 15?' do not typically yield actionable insights because they focus on a very specific value rather than on understanding model behaviour under variable conditions. A more practical

approach might reframe the problem in a classification term, asking 'Under what conditions does the model predict values below 14?' This reframing not only makes the question more specific, but also more actionable, as it guides the development of explanation methods by examining how changes in input affect the threshold-based classification. Through this transformation, forecasting models are actually treated as classification problems, which facilicates the design of explanation methods.

Moreover, the emphasis of this thesis is on multivariate time series because they encapsulate more information than univariate series, offering richer insights for prediction. Modern deep learning models are adept at extracting and leveraging this information for high-performance predictions. However, these models often lack interpretability. In addition, the explanation methods designed for multivariate time series establish a general framework that is also adapted for univariate time series. Therefore, this thesis is dedicated to improving the interpretability of deep learning models for MTSC.

In the XAI domain, time series models have not received as much attention as they deserve. A key reason for this could be attributed to the inherent characteristics of time series data. Typically, time series are represented by a sequence of discrete values, the meanings of which are often not immediately clear to humans. This contrasts with tabular data, where each feature usually has a specific and well-defined physical meaning. Similarly, in language models, each input word is easily understood, and in image models, pixels often represent recognisable objects. However, in time series data, the meaning of each value might be less apparent than that of images or words.

The inherent obscurity of time series data presents significant challenges

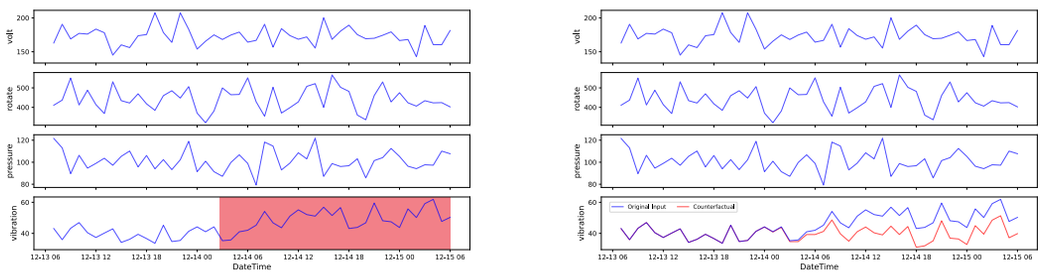
in providing effective explanations for MTSC. The main objective of XAI is to provide explanations that are not only comprehensible to humans, but also align with their individual requirements. The diverse knowledge backgrounds and varying needs of users imply that different individuals may favour different types of explanation. Given the subjective nature of the term 'explanation', it becomes essential to clearly identify and understand the preferences of the target audience prior to developing explanatory methods. Effective explanations are those that resonate with and meet the specific needs of the target audience. Consequently, a crucial step in advancing XAI within the MTSC domain involves pinpointing the needs and preferences of the target audience, ensuring that the explanations provided are comprehensible and effective for diverse stakeholders.

1.2 Tailoring Explanations for Diverse Stakeholders

In Spring 2023, the UK government published its policy paper titled "A pro-innovation approach to AI regulation". In it, they set out five cross-cutting principles that will underpin the UK's approach to AI regulation (Roberts et al., 2023). One of these principles emphasises the need for appropriate transparency and explainability, which involves making the workings of AI systems understandable to their users. However, a critical question arises: Which forms of explanations are most effective and should be provided? The challenge lies in the subjective nature of the explanations; different stakeholders require different types of explanation. For example, a domain expert might need technical explanations involving model architecture and algorithmic decision-making processes. Nevertheless, for ordinary people,

simpler and more intuitive explanations are necessary, which translate complex AI decisions into understandable language and practical implications. Therefore, it is necessary to develop a variety of explanation methods tailored to meet the diverse needs of various stakeholders. Given the need for varied types of explanations, post-hoc explanation techniques become particularly valuable. These techniques are capable of providing a range of explanations to suit different audiences, in contrast to self-explainable approaches, which are typically limited to a single type of explanation predetermined during the model construction stage. Therefore, this thesis concentrates on developing post-hoc explanation methods that cater to a spectrum of explanatory needs. Among various types of explanations, including causal explanations, example-based explanations, feature importance, and counterfactual explanations, the latter two emerge as the predominant categories (Barredo Arrieta et al., 2020). Feature importance explanations are exceptionally beneficial for AI experts, providing in-depth insights into which features most significantly influence model outcomes. In contrast, counterfactual explanations hold particular value for general audiences by demonstrating how changes in inputs could lead to different model predictions. Therefore, to meet the various requirements of different audiences, this thesis focuses on feature importance and counterfactual explanations, both of which are illustrated in Figure 1.2.

Feature Importance Explanations: Feature importance is among the most common types of explanation, where the most important features are identified for a certain input. In the context of MTSC, values in certain time steps can be regarded as features, similar to pixels in images or features in tabular data (Crabbé and Van Der Schaar, 2021; Ismail et al., 2020). Therefore, in feature importance explanations the most important time series subsequences are highlighted, shown in Figure 1.2a. These ex-



(a) (b)

Figure 1.2: (a) Feature Importance Explanation: Highlighting the most important time series subsequences. (b) Counterfactual Explanation: Demonstrating how an alternative past vibration level, as indicated by the red line, would lead to a different classifier prediction.

planations are particularly beneficial for stakeholders, who may use them for feature selection to enhance model performance or to ensure the model's alignment with established scientific knowledge. For example, in the clinical area, time series data form a crucial basis for developing early disease detection models. However, clinical time series data are often complex and derived from various sources. For model constructors, especially those without a medical background, feature importance explanations can be invaluable in identifying and eliminating non-essential features, helping them reduce model complexity and potentially improve performance. Beyond the raw time series data shown in Figure 1.2a, some time series also incorporate some static features, such as age and gender, which can significantly influence a model's behaviour. Understanding the importance of these features is also crucial as it enables users to gain a comprehensive view of the factors driving the model predictions. For example, in clinical settings, recognising the significance of age and gender in conjunction with physiological measurements can greatly enhance early disease detection models. By tailoring these models to specific demographic groups, both the accuracy and relevance of the predictions can be improved.

Counterfactual Explanations: Counterfactual explanations represent another well-known type of explanation (Ates et al., 2021a; Delaney et al.,

2021a). Counterfactual explanations aim to answer how specific changes in input variables could lead to a different desired outcome from a predictive model. As shown in Figure 1.2b, should the past vibration level be represented by the red line instead of its original state, the classifier will produce a prediction that differs from that based on the original input. By illustrating 'what-if' scenarios, these explanations help users understand the conditions under which the outcome would change, enabling them to make informed decisions based on the insights of the model. This kind of explanation is beneficial to people, particularly those who may not have in-depth technical expertise, as it provides clear and actionable insights (Poyiadzi et al., 2020). For example, in applications such as early disease detection, imagine that an automated diagnostic system predicts the development of a disease in the future. Along with this prediction, a counterfactual explanation reveals that if blood sugar levels had been reduced to a certain threshold during a specified period, the model would have predicted that the disease would not occur. Armed with such insights, physicians can take proactive measures to manage blood sugar levels, potentially preventing the development of the disease.

1.3 Gaps and Objectives

MTSC has a wide range of real-world applications, and deep learning techniques have significantly advanced high-performance solutions in this area. However, a critical limitation is their lack of interpretability, which restricts their deployment in safety-critical fields such as finance and healthcare. XAI has emerged as a crucial domain within AI, with the objective of demystifying these models. However, applying current XAI methods to MTSC presents challenges that this thesis aims to address.

- Time series data are characterised by their temporal dependency, meaning that each data point is in relation to its chronological predecessors and successors. This aspect fundamentally differs from tabular data, where features are generally assumed to be independent. Therefore, most of the current feature importance explanation methods, which often overlook these temporal dependency, are not applicable for MTSC. These methods tend to generate unrealistic time series that are not aligned with the data encountered by the classifier during its training phase. Therefore, the predictions made on these unrealistic time series are not reliable and cannot reflect what the model has learnt in the training stage, resulting in unfaithful explanations. This is known as the out-of-distribution problem in the XAI domain (Hase et al., 2021; Kim et al., 2020), yet it is overlooked in the context of MTSC explanations.

- Another significant challenge arises specifically with some proxy explanation methods, such as LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016), a well-known proxy explanation framework. Recent research has identified a critical issue regarding the stability of the explanations provided by LIME. This instability means that the explanations generated for the same instance by the same settings can vary inconsistently, undermining the trust users place in these explanations (Zhou et al., 2021; Visani et al., 2020). Although some researchers suggest increasing the number of neighbours or optimising hyperparameters within the explanation framework to enhance stability, these strategies appear insufficient in the context of MTSC. The underlying reasons for this instability in the context of MTSC remain unclear, which presents a significant hurdle in applying this method in this field.

- The final challenge addressed in this thesis targets the feasibility and plausibility of counterfactual explanations provided in the context of MTSC. Various counterfactual explanation methods have been proposed, some specifically designed for MTSC (Delaney et al., 2021a; Ates et al., 2021a). However, these methods primarily concentrate on minimising alterations to the original input to achieve a desired classifier's outcome. This type of approach often results in the generation of unrealistic counterfactual explanations. These explanations, while technically valid to change model's outcomes, fail to offer meaningful insights or actionable intelligence. This is because they do not adequately consider the complex, real-world dynamics and dependencies inherent in time series data. Consequently, such counterfactuals may not be representative of feasible or realistic scenarios, limiting their utility in practical applications.

The main objective of this thesis is to develop post-hoc techniques in the context of MTSC to provide various kinds of explanation, including feature importance and counterfactual explanations, with the aim of addressing the diverse interpretability requirements of various stakeholders. The gaps identified above serve as a road map for a targeted advancement of XAI within the MTSC domain. In response to these gaps, the research questions for this thesis are outlined as follows:

- *RQ1*: Is it possible to generate within-distribution samples for the MTSC problem to improve the performance of feature importance explanation methods?

- *RQ2*: What are the causes of the unstable explanations provided for MTSC by LIME, and does generating within-distribution samples help to stabilise the explanations?

- *RQ3*: How can we ensure that counterfactual explanations in MTSC align with the training data distribution, thus enhancing their feasibility and plausibility?

Informed by the motivations and research questions previously outlined, the central aim of this thesis is articulated as: to advance the development of XAI within the MTSC context through generative models with the aim of meeting the varied requirements of different stakeholders by providing feature importance and counterfactual explanations. To achieve this overarching goal, three primary research objectives have been established.

- *Objective 1*: to construct a comprehensive framework that effectively incorporates the temporal dependencies inherent in time series data, with the aim of creating within-distribution samples for accurate feature importance evaluation in MTSC scenarios.
- *Objective 2*: to investigate the causes of the instability of the explanations provided by LIME in the context of MTSC. Based on this analysis, the objective is to design and implement solutions that ensure the stability and consistency of these explanations.
- *Objective 3*: to create a robust framework for MTSC that is capable of generating plausible and practical counterfactual explanations. The focus here is to enhance the usefulness and applicability of these explanations in real-world decision-making scenarios.

1.4 Contributions

The contributions of this thesis are to advance the development of XAI in the context of MTSC, providing various types of explanation to meet the

needs of different stakeholders. The types of targeted explanation are feature importance and counterfactual explanations. The primary approach of this thesis focuses on examining the potential shortcomings of state-of-the-art explanation methods when applied within the MTSC and adapting them to be more suitable for this specific domain. The key contributions of this thesis include:

- **A Model-agnostic Post-hoc Framework for Feature Importance Explanations**: A novel model-agnostic post-hoc framework is established to explain MTSC through feature importance. There are two innovations within the proposed framework. First, a generative model is designed and used to generate perturbation samples within the training data distribution, which are used to accurately evaluate the importance of features. Second, a novel heuristic optimisation method is designed for the MTSC problem to search for the most important features more efficiently.

- **Uncovering the Causes of Unstable Explanations in LIME**: This thesis is the first work to distinctly highlight that the OOD issue is a primary factor contributing to the instability of the explanations provided by LIME. The impact of OOD on the stability of the explanation has not been investigated in the existing literature. This thesis provides a comprehensive analysis, revealing how the OOD problem can significantly affect the stability of LIME's explanations.

- **Stable LIME Explanation Framework**: A stable LIME framework, SEGAL (Stable Explanations using a Generative Model and an Adaptive Weighting Method for LIME), is developed. The proposed SEGAL framework has two innovations. First, a generative model is employed to generate meaningful neighbours. Second, an

adaptive weighting method is designed in which the hyperparameters are easier to tune than those of the classic approaches.

- **Plausible Counterfactual Explanations**: An innovative counterfactual explanation method is developed, with a primary emphasis on enhancing the plausibility of explanations to ensure their relevance and applicability. To achieve this, we have devised two distinct approaches. The first approach uses a Gaussian mixture model to approximate the density distribution of the data, enabling the generation of counterfactuals within the regions of highest density. This ensures that the counterfactuals are not only plausible but also closely aligned with the most probable scenarios in the data distribution. The second approach employs a Generative Adversarial Network (GAN) framework, focusing on producing counterfactuals that are indistinguishable by the classifier. This method leverages the power of GAN to create realistic and plausible counterfactuals.

The above contributions are part of or included in the following list of works completed during the PhD studies:

- Han Meng, Christian Wagner, Isaac Triguero. Feature Importance Identification for Time Series Classifiers. 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 2022, pp. 3293-3298, DOI: 10.1109/SMC53654.2022.9945205.
  The content of this paper is covered in Chapter 3.

- Han Meng, Christian Wagner, Isaac Triguero. Explaining Time Series Classifiers through Meaningful Perturbation and Optimisation. Information Sciences, 2023. DOI:10.1016/j.ins.2023.119334.
  The content of this paper is covered in Chapter 3.

- Han Meng, Christian Wagner, Isaac Triguero. An Initial Step Towards Stable Explanations for Multivariate Time Series Classifiers with LIME. 2023 IEEE International Conference on Fuzzy Systems (FUZZ), Incheon, Korea, Republic of, 2023, pp. 1-6, DOI: 10.1109/FUZZ52849.2023.10309814.

  The content of this paper is covered in Chapter 4.

- Han Meng, Christian Wagner, Isaac Triguero. SEGAL Time Series Classification - Stable Explanations using a Generative Model and an Adaptive Weighting Method for LIME. Neural Networks. 2024, DOI:10.1016/j.neunet.2024.106345.

  The content of this paper is covered in Chapter 4.

- Han Meng, Christian Wagner, Isaac Triguero. A plausible counterfactual explanation method for multivariate time series classifiers. (in preparation).

  The content of this paper is covered in Chapter 5.

1.5 Thesis Organisation

This thesis is organised as follows:

- Chapter 2 delves into the existing prominent post-hoc explanation methods within the realm of XAI. It highlights the limitations and shortcomings of these methods when applied to MTSC, along with a discussion of the specific challenges encountered in explaining MTSC models. This sets the stage for subsequent chapters by establishing the current state of the field.

- Chapter 3 details the proposed post-hoc model-agnostic framework

for feature importance explanations in MTSC. This chapter details the architecture of the generative model and the innovative greedy-based heuristic search method. In addition, this chapter discusses the strengths and limitations of the proposed framework.

- In Chapter 4 the proposed SEGAL framework is presented, which covers the neighbour generation approaches utilising the specially designed generative model and introduces an adaptive weighting approach to ease the challenges associated with hyperparameter tuning. Moreover, a significant focus is placed on elucidating how the OOD problem contributes to unstable explanations, offering deep insights into this problem.

- Chapter 5 introduces novel counterfactual explanation methods designed for MTSC. It describes the methodologies used to enhance the meaningfulness of counterfactual explanations, including maximising the data density and generating indistinguishable counterfactuals. The practical utility of these counterfactual explanations in real-world scenarios is presented.

- Chapter 6 summarises the key contributions of this thesis, reflecting on its impact in the field of XAI for MTSC. It also identifies current limitations and outlines potential avenues for future research.

Chapter 2

Literature Review

Technological advancements have significantly advanced modern artificial intelligence, notably deep learning, which excels at identifying complex patterns in data for various tasks. However, its deployment in safety-critical areas is impeded by a lack of interpretability. In these areas such as finance and healthcare, where MTSC has a wide range of applications, deep learning-based models offer promising solutions, but their interpretability remains a challenge. Although many explanation methods have been developed in the XAI domain, applying them to MTSC problems presents several challenges that need to be addressed.

This chapter lays the foundation for understanding the subsequent content of this thesis. Specifically, Section 2.1 introduces the background of XAI, covering its significance within the AI community, foundational concepts, various categories of explanation methods, and the types of explanations provided. Section 2.2 defines the MTSC problem that this thesis focuses on. Section 2.4 discusses the state-of-the-art feature importance explanation methods and their limitations in the context of MTSC. Section 2.5 covers the desirable properties of counterfactual explanations, the current

counterfactual explanation methods proposed in the literature for MTSC, and their limitations.

2.1 eXplainable Artificial Intelligence

XAI seeks to demystify the models' behaviour, making them transparent and comprehensible to humans (Barredo Arrieta et al., 2020)[1]. This field has gained momentum as AI applications have become more widespread and complex. In the early days of AI, systems were relatively simple with logic that could be easily understood. However, the emergence of deep learning has ushered in an era of more intricate models, amplifying the need for clarity about how these models operate, especially in critical areas such as healthcare and finance. Understanding the rationale behind these complex models is crucial for users, as they need explanations to gain trust and avoid potential risks (Barredo Arrieta et al., 2020). In recent years, XAI has emerged as a vibrant area of research, drawing considerable attention in the academic community.

As the field of XAI has evolved, numerous concepts have been introduced that may initially seem daunting to newcomers, such as interpretability, explainability, transparency, understandability, etc. Currently in the tge XAI domain, the meaning of these terms still varies across different literature, although some agreement is appearing.

- Interpretability refers to the extent to which a human can understand the rationale behind an AI system's decisions. For example, in an AI

[1]Model behaviour refers to how a model processes input data to produce outputs or decisions. In this thesis, we specifically focus on the model's responses to samples that fall within the same distribution as the training data, which reveals the knowledge that the model has learnt during the training phase.

medical diagnosis, interpretability might be demonstrated through a model that highlights the symptoms or test results that influenced its diagnosis of a disease the most. This helps clinicians understand why the AI made a particular decision, potentially helping them in trust building and decision-making processes.

- Explainability extends beyond interpretability by not only revealing the influences behind decisions but also describing the mechanics of the model in terms that are relatable to human understanding (Guidotti et al., 2018). Continuing with the medical example, an explainable AI model would not only highlight influential factors, but would also provide a simplified description or visualisation of how these factors interact to lead to a specific diagnosis. This could include showing how different symptom weights contribute to the final output, thereby making the AI's process more accessible to healthcare professionals without deep technical expertise.

- Transparency, on the other hand, refers to the inherent openness of a model in revealing its workings directly (Lipton, 2018). A transparent model in the AI domain could be a decision tree in which each decision node and pathway are clearly visible and understandable to the user. For example, a financial lending AI that uses a decision tree can show exactly how it assesses creditworthiness based on criteria such as credit score, income, and debt levels, and each decision point transparently contributes to the final decision to approve or deny a loan.

In this thesis, our focus will be on interpretability and explainability. Our objective is to develop techniques that elucidate the inner workings of models in a manner that is readily accessible to users, enabling them to under-

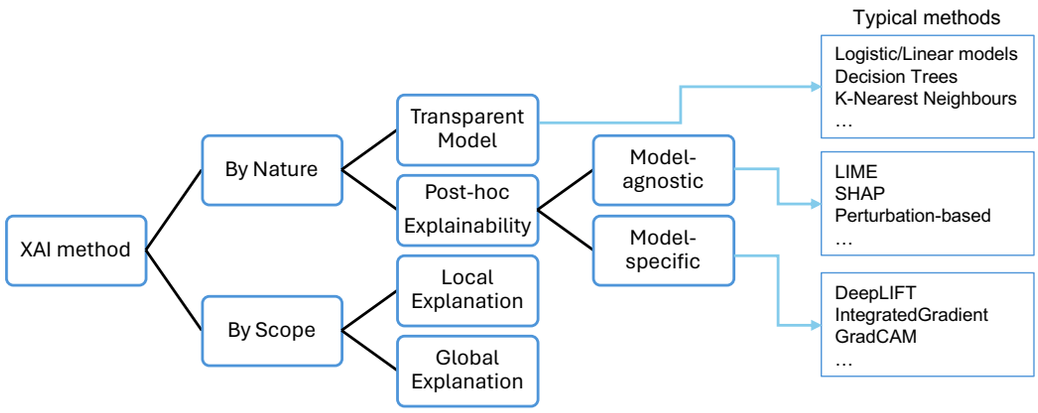


Figure 2.1: The category of the approaches to obtain the explanations for machine learning models

stand the mechanisms underpinning model decisions with ease.

The foundational concepts introduced above pave the way for the advancement of XAI development. Currently in the literature, the methods developed to obtain explanations can be categorised based on various certeria, which can be shown in Fig. 2.1. First, explanations for models can be obtained through either post-hoc techniques or self-explainable models.

- *Post-hoc techniques*: Implemented after a model has been trained and made its predictions; these methods retrospectively analyse and elucidate the decision-making process.
- *Self-explainable models*: Incorporating explanatory capabilities during the model construction phase, these models are inherently transparent, designed with the purpose of being understandable from the outset. This intrinsic transparency allows self-explainable models to provide a global perspective on their decision-making processes. They enable a comprehensive understanding of the model's operational logic over a wide array of inputs, thereby offering a holistic view of its behaviour and principles.

Furthermore, post hoc interpretability can be obtained through either model-

agnostic or model-specific approaches.

- *Model-agnostic methods*: These are versatile techniques that can be applied to various models, regardless of their underlying architecture. They offer flexibility in explaining different models without the need to understand their internal mechanics.

- *Model-specific methods*: Tailored to specific model architectures, these methods leverage the inherent characteristics of a model to provide explanations. For instance, some gradient-based explanation techniques are applied exclusively to models with accessible gradients (Simonyan et al., 2014).

Explanation techniques can also be categorised based on the scope of their explanations, specifically whether they aim to provide local or global explanations.

- Local explanation methods: These methods focus on individual predictions made by a model, providing insights into why a model made a specific decision for a specific instance.

- Global explanations: These explanations aim to provide an overview of what a model has learnt overall, rather than focusing on individual predictions.

As noted above, a central tenet of XAI is the comprehensibility of the explanations it provides. The effectiveness of these explanations is largely dependent on their ability to be understood by humans. However, the diversity in AI stakeholders' backgrounds and needs leads to varied perceived utility in respect to the same explanations. In other words, various types

of explanations generated by these methods cater to different interpretation needs and preferences (Barredo Arrieta et al., 2020). This variability underscores the importance of tailoring explanations to suit the specific understanding and context of the audience, ensuring that the insights offered by AI systems are accessible and meaningful to all users, regardless of their expertise or familiarity with the underlying technology. Currently in the literature, the explanation types provided are broadly categories into several types (Barredo Arrieta et al., 2020):

- *Feature Importance*: This type emphasises the features that significantly influence the predictions of a model. One prominent method used to quantify this influence is SHAP (SHapley Additive exPlanations), which is based on Shapley values from cooperative game theory. It provides a robust way to allocate "payouts" (i.e., the prediction output of the model) to individual "players" (i.e., features) based on their contribution to the cooperative game (i.e., the predictive model). On the other hand, LIME (Local Interpretable Model-agnostic Explanations) offers a different approach. LIME aims to explain the predictions of any classifier or regression model by approximating it locally with an interpretable model, such as a linear model or a decision tree. By perturbing the input data and observing the changes in predictions, LIME generates explanations that detail how individual features affect the prediction at a specific instance (Ribeiro et al., 2016).

- *Counterfactual Explanations*: These explanations illustrate how alterations in input values could lead to different predictions, offering insights into the model's response to changes in input. A notable counterfactual explanation was introduced by (Wachter et al., 2017), who suggested to search for the minimal changes required to the orig-

inal input to achieve the desired output.

- *Example-Based Explanations*: By using specific data points as benchmarks, these explanations clarify the model's behaviour through understandable examples. Example-based explanations can be derived by identifying relevant instances within the existing data set (van der Waa et al., 2021), or by employing methods designed to generate such explanations (Kim et al., 2016).

- *Causal Explanations*: Aimed at elucidating the cause-and-effect relationships within the data that the model uses to make decisions, these explanations strive to provide a deeper understanding of the model's logic. Key contributions in this area include the work by Pearl (Neuberg, 2003), which lays the theoretical foundations for causal reasoning in AI, and the more recent advancements by Peters et al. (Peters et al., 2017), who explore the use of causal methods to interpret complex machine learning models.

Among these explanations, feature importance and counterfactual explanations have received significant attention because of their straightforwardness and efficacy, satisfying the needs of a broad spectrum of users. Feature importance explanations are particularly advantageous for AI specialists and domain experts, offering granular insights into the features that play a pivotal role in influencing the model's predictions, which can help me to feature selections, simply the models, and improve models' performance. For example, in clinical settings, time series data serves as a foundational resource for data scientists to develop models for early disease detection. However, real-world clinical time series data is intricate, and it contains series from diverse sources. For data scientists, who usually lack medical background, feature importance explanations could assist them in filtering

out non-essential features, reducing the model's complexity, and enhancing the model's performance.

On the other hand, counterfactual explanations play a critical role in demystifying AI for users of AI products in a broad sense. By presenting hypothetical scenarios where slight modifications to input values result in different outcomes, these explanations provide intuitive and relatable insights into the models' operations. For example, in the same early disease detection application, physicians desire insights beyond only model predictions. They aim to seek treatment to prevent disease progression. Instance-based explanations can offer specific indicators. For instance, the elevated blood sugar in the last 24 hours leads to the disease (factual explanation). If the blood sugar reduce to a certain level in the past 24 hours, the model predicts that the disease will not develop (counterfactual explanation). Armed with such explanations, physicians can consider proactively managing blood sugar levels to prevent disease development.

In this thesis, we aim to develop post-hoc techniques that provide feature importance and counterfactual explanations. Having explored the critical principles of XAI, we now direct our focus to MTSC, the central research area of this thesis. The following section presents a comprehensive review of MTSC, setting the stage for the detailed discussions and analyses that follow.

2.2 Multivariate Time Series Classification

A time series is a sequence of real values recorded in temporal order. A time series can be univariate, where the values are collected from only one source, or multivariate, where the real values are collected from multiple

sources. An example of an Multivariate Time Series (MTS) can be seen in Fig. 2.2. The time series is indicated by $\boldsymbol{x} \in \mathbb{R}^{D \times T}$, where D is the number of variables (or the number of sources from which the data are collected; for univariate time series, $D = 1$) and T is the total number of time steps.

When talking about time series models, researchers often first consider forecasting, which involves predicting future values based on historical data, such as the price of stock tomorrow (Sezer et al., 2020). However, time series classification, which might be less well-known than forecasting, involves categorising time series data into specific classes. For example, Fig. 2.2 shows an MTSC problem that classifies a time series with three sequences into class "2". Similar to time series forecasting, MTSC is widely applied in various real-world scenarios. Examples include diagnostic systems that interpret EEG data[2] to assess brain conditions, and automated trading systems that classify future stock price movements based on historical trading data. This thesis concentrates on MTSC, but the methods proposed here are also applicable to forecasting by transforming original forecasting problems into classification tasks. For example, in (Assaf et al., 2019), the forecast target is grouped into six classes, and then a classifier is trained on this classification data set. In this thesis, the classifiers to be explained are denoted by $f(c|\boldsymbol{x})$, which predicts the probability of a given input time series $\boldsymbol{x}$ belonging to a class c. The label assigned to $\boldsymbol{x}$, which is denoted as $f(\boldsymbol{x})$, is one corresponding to the highest probability.

Machine learning has played a crucial role in the development of MTSC. For example, decision trees (Douzal-Chouakria and Amblard, 2012), k-nearest neighbours (Lee et al., 2012), and support vector machines (Kampouraki et al., 2008) stand out as promising and have shown great success. These

[2]EEG, or Electroencephalography, captures the brain's electrical activity in a temporal sequence, making it inherently a type of time series data.

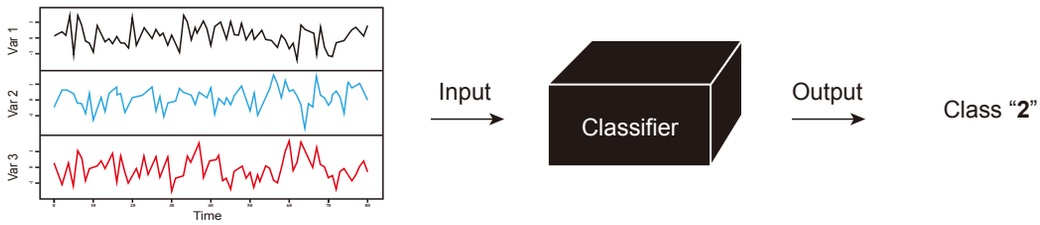


Figure 2.2: An output, $f(\boldsymbol{x}) =$ "2", is produced by a classifier for an MTS. This MTS has 3 variables, and the length of these sequences is 80. Therefore, the input is denoted by $\boldsymbol{x} \in \mathbb{R}^{3\times 80}$ and has 240 features.

methods are simple and easy to understand. However, the success of these approaches often hinges on effective data preprocessing or the use of sophisticated distance metrics tailored to time series data (Geler et al., 2020). Recently, deep learning, which is particularly effective in capturing complex patterns within the data, has been proven to be effective in MTSC and has been widely adopted (Ismail Fawaz et al., 2019). However, these approaches are often black boxes, lacking the explanations necessary to understand their results. This lack of transparency limits their applications in critical sectors where model users require reliable explanations to build trust and facilitate decision-making processes (Barredo Arrieta et al., 2020). Therefore, this thesis focuses on explaining deep learning based models.

2.3 The Out-Of-Distribution problem in AI

The OOD problem in AI occurs when a model encounters data that do not align with the distribution of its training data. For example, this issue arises when an image of a human is fed into a classifier trained to differentiate between cat and dog images. Predictions made in such scenarios are often unreliable and not meaningful because the model lacks the necessary context to correctly interpret the human image, which it has never been trained to recognise. Thus, human images are considered OOD for

this cat-dog classifier. This type of OOD problem can be easily recognised. However, the OOD issue can be more difficult to identify in some scenarios. For example, if a model is trained on data obtained from Asian individuals, samples collected from African individuals may be considered OOD. In some scenarios, the difference between the data collected from Asian individuals and that from African individuals is difficult to judge for humans. A mathematical definition of the OOD problem can be formulated as follows:

$$\sup_{\boldsymbol{x}\in\mathcal{X}} |p_{train}(\boldsymbol{x}) - p_{test}(\boldsymbol{x})| > \epsilon \tag{2.1}$$

$p_{\text{train}}(\boldsymbol{x})$ represent the probability distribution of the data set on which the model is trained, and $p_{\text{test}}(\boldsymbol{x})$ denote the probability distribution of the dataset on which the model is tested or to which the model is exposed during the explanation process. A probability distribution of a data set describes the way in which data points within that data set are spread or distributed across different possible values. This distribution captures the likelihood of each possible value or range of values occurring within the data set. When these distributions differ significantly, as quantified by a threshold ϵ, the model faces an OOD problem. This discrepancy can lead to poor performance as the model encounters data points that are not represented in the training distribution, making accurate and reliable predictions challenging.

In the XAI domain, the OOD problem often arises because many explanation processes, especially post-hoc explanation methods, involve altering the input data, such as perturbing certain features (Hase et al., 2021; Qiu et al., 2021). These perturbations might disrupt the inherent dependencies within the input data, resulting in samples that are OOD. For example, existing research often perturbs time series data by replacing certain values with zeros. However, in practical scenarios, some values, such as blood

pressure, cannot be zero. This perturbation creates OOD input data as it does not reflect any physiological condition and therefore deviates significantly from any training distribution to which the model may have been exposed. The explanations provided by the OOD samples can be misleading because they may not accurately reflect the decision-making process of the model that is obtained in the training stage. When making predictions on OOD samples, the model may rely on partial and incomplete information, leading to conclusions that are not grounded in the same reality as the training data. This can cause significant discrepancies between the learnt behaviour of the model in the training stage, potentially leading to incorrect interpretations and decisions based on these explanations.

Currently, the literature presents several approaches to address the OOD problem in the XAI domain. The first approach involves retraining the model with perturbed samples that are used in the explanation process (Hase et al., 2021; Qiu et al., 2021). These samples are added to the training data and the model is subsequently re-trained on this enriched dataset. This strategy ensures that the samples used in the explanation process are aligned with the training data. However, this solution faces significant limitations. First, retraining large models can be computationally prohibitive and sometimes impractical. Second, including such samples might compromise the overall performance of the model by introducing noise or irrelevant features. The second approach involves using a generative model to learn the data distribution and subsequently generate within-distribution samples for the explanation process (Lang et al., 2023a). Although this method has been successfully applied in image models (Chang et al., 2019) and language processing models (Kim et al., 2020), it has not yet been adapted for time series data. A primary limitation is the lack of effective generative models specifically designed for time series data, which presents unique

challenges in accurately capturing temporal dependencies and dynamics.

2.4 Feature Importance Explanations

Feature importance explanation methods are one of the most intuitive approaches to explain complex black-box classifiers (Ismail et al., 2020). These methods involve creating an importance heatmap, assigning scores to each feature based on its influence on the classifier's output. Within this thesis, a value at a certain time step is regarded as one feature. For example, $\boldsymbol{x}_{i,t}$ is the feature that represents the value of the i-th variable at time t. Therefore, the MTS shown in Fig. 2.2 has 240 features. An example of a classic feature importance explanation for an MTSC is shown in Fig. 2.3. Currently, methods that provide classic features importance explanations for black-box classifiers can be roughly classified into three groups: *gradient-based* methods, *perturbation-based* methods, and *proxy explanation* methods.

2.4.1 Gradient-based Methods

Gradient-based methods operate on the premise that a significant change in output will be triggered by the small variation of an important feature. The importance of a feature is assessed by the gradient of the probability of a specific class in the output with respect to that feature. The 'gradient' in this context refers to the rate of change of the class prediction probability with respect to the change in the input feature. A larger gradient indicates that slight variations in the feature significantly impact the model's output, suggesting the feature's higher importance.

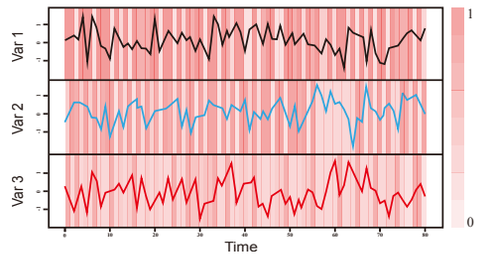


Figure 2.3: A classic feature importance explanation, where each feature is assigned with an importance score.

Popular gradient-based methods include Simple Gradient (SG) (Simonyan et al., 2014), Integrated Gradient (IG) (Sundararajan et al., 2017), SmoothGrad (Smilkov et al., 2017), Grad-CAM (Gradient-weighted Class Activation Mapping) (Selvaraju et al., 2017a), Layer-wise Relevance Propagation (LRP) (Bach et al., 2015), etc. These methods can provide explanations very quickly, but their explanations might be misleading (Adebayo et al., 2018). SG calculates the partial derivatives of the output with respect to each input feature, providing a gradient vector that indicates the importance of each feature, which can be expressed as:

$$SG(\boldsymbol{x}_i) = \frac{\partial f(c|\boldsymbol{x})}{\partial \boldsymbol{x}_i} \tag{2.2}$$

where $SG(\boldsymbol{x}_i)$ denotes the feature importance of the ith feature, $f(c|\boldsymbol{x})$ is the probability of class c given input vector $\boldsymbol{x}$, and $\frac{\partial}{\partial \boldsymbol{x}_i}$ represents its partial derivative with respect to feature $\boldsymbol{x}_i$. This method is both straightforward and efficient for deep learning models, as gradients can be easily obtained by backpropagation. However, it may fail in some scenarios, such as model saturation (Shrikumar et al., 2017). To elucidate this problem, let us consider a simplified model represented by the function $y = \max(0, 1-x_1-x_2)$. This function is served as an illustrative example to demonstrate the phenomenon known as 'model saturation'. For this simplified model, as long as $x_1 + x_2 > 1$, the output of this model remains constant (at zero), resulting in a zero gradient with respect to these inputs.

IG is a method designed to overcome this limitation faced by SG by considering the path integral of the gradients along a straight path from a baseline input to the actual input. The mechanism of IG can be formulated as follows:

$$IG(\boldsymbol{x}_i) = (\boldsymbol{x}_i - \boldsymbol{x}_i^{\text{baseline}}) \times \int_{\alpha=0}^{1} \frac{\partial f(c|\boldsymbol{x}^{\alpha})}{\partial \boldsymbol{x}_i} d\alpha \tag{2.3}$$

Here, $IG(\boldsymbol{x}_i)$ denotes the importance of feature $\boldsymbol{x}_i$; $\boldsymbol{x}_i^{\text{baseline}}$ is the baseline value for feature $\boldsymbol{x}_i$, typically set to a specific value in the literature; $f(c|\boldsymbol{x}^{\alpha})$ represents the model's prediction probability for class c given the input vector $\boldsymbol{x}^{\alpha}$. This input vector $\boldsymbol{x}^{\alpha} = \boldsymbol{x}_i^{\text{baseline}} + \alpha \times (\boldsymbol{x}_i - \boldsymbol{x}_i^{\text{baseline}})$ linearly interpolates between the baseline input and the actual input. The integral $\int_{\alpha=0}^{1} d\alpha$ calculates the accumulation of gradients along the path from the baseline to the actual input, effectively capturing the contribution of the feature throughout the whole path. Therefore, the IG needs a baseline to generate explanations. The authors of the work (Sundararajan et al., 2017) set the baseline to 'zeros' input, which might risk feeding classifiers with inputs that classifiers did not observe during the training stage, resulting in unreliable predictions. Furthermore, different baselines will result in different explanations, but no guidance has been developed to choose a suitable baseline (Sturmfels et al., 2020).

Another limitation of the SG method is its sensitivity to local variations and noise in the input, which can result in explanations that are difficult to interpret because of the high variance in the gradient information. To address this issue, SmoothGrad (Smilkov et al., 2017) was proposed as a technique that improves the clarity and reliability of gradient-based explanations. SmoothGrad works by averaging the gradients of the model's output with respect to the input over multiple instances of the input with

added noise. Specifically, it generates a set of noisy versions of the input by adding small random perturbations, and then computes the gradient for each noisy input. The final explanation is the average of these gradients, which tends to smooth out the noise and highlight more consistent patterns in the input's influence on the output.

LRP (Bach et al., 2015) operates by backpropagating the prediction output through the layers of the network, distributing the prediction back to the input features in a manner that reflects their contribution to the final decision. This is achieved through a set of rules that guide the redistribution of relevance scores from the output layer back to the input layer, effectively decomposing the prediction into contributions attributable to each input feature. Mathematically, LRP can be formulated as redistributing the output $f(\boldsymbol{x})$ back to the input features $\boldsymbol{x}_i$ by iteratively applying the LRP rules across layers, quantifying the relevance R_i of each input feature as:

$$R_i = \sum_j \frac{a_i w_{ij}}{\sum_i a_i w_{ij}} R_j \tag{2.4}$$

where a_i represents the activation of neuron i, w_{ij} is the weight connecting neuron i to neuron j, and R_j is the relevance of neuron j in the subsequent layer. The formula ensures that the relevance propagated back to the input layer accurately reflects the contributions of individual features to the output. It provides a detailed, layer-by-layer breakdown of how input features contribute to the network's output, offering deep insights into the model's decision-making process. However, the propagation of relevance scores can become opaque or difficult to follow, particularly in very deep or complex models, which may reduce the clarity of the explanations. Moreover, the method assumes that all layers contribute equally to the model's decision-making, which might not always hold true, potentially oversimplifying the

contributions of more complex interactions within the network.

Grad-CAM technique (Selvaraju et al., 2017a), specifically tailored for convolutional neural networks (CNNs), leverages the gradients of a target output (e.g., the score of a specific class in classification tasks) relative to the activations of the final convolutional layer. This process generates a coarse heatmap that identifies the input regions most influential for the prediction of the model. A significant advantage of Grad-CAM is its ability to accurately highlight and visualise the regions within the input that CNN considers crucial to making its decisions. However, Grad-CAM's specificity to convolutional network architectures limits its applicability to models of other types, restricting its utility in non-CNN contexts. Additionally, Grad-CAM provides explanations based on the assumption that the final convolutional layer effectively captures the essential spatial information for the task at hand. This assumption may not always be valid across different CNN architectures or for layers other than the last convolutional one, potentially impacting the relevance and accuracy of the generated heatmaps.

2.4.2 Perturbation-based Methods

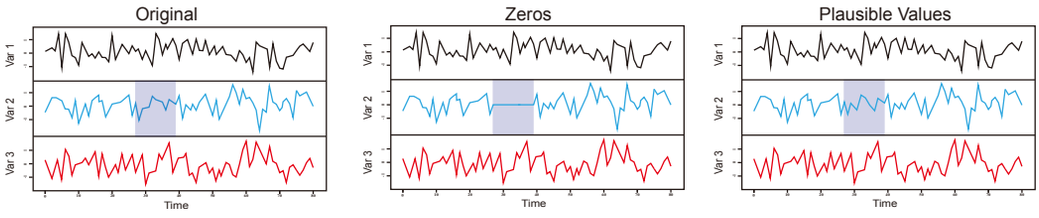
Gradient-based methods are specifically designed for models where gradients can be easily computed, such as deep learning models. This characteristic renders them model-specific, as their applicability is contingent upon the availability of gradient information. In contrast, perturbation-based methods offer a wider scope of application. They perturb input features and subsequently measure the resultant changes in the classifier's output. They operate under the assumption that perturbations of certain important features will lead to significant changes in the output.

When using perturbation-based methods, one of the key points is how to perturb features. Currently, in the context of MTSC, the features are perturbed by replacing them with zeros (Assaf et al., 2019; Bento et al., 2021a) or with the weighted average of their neighbouring features (Crabbé and Van Der Schaar, 2021). However, these perturbation operations might generate inputs that the classifier never observes in the training stage, as shown in Fig. 2.4a. The predictions made on these inputs are unreliable and cannot reflect what the model has learnt from training data sets, which is known as the OOD problem.

In the context of time series, the OOD problem may not be as obvious to humans as it is in image problems, as shown in Fig. 2.4b. For example, from Fig. 2.4b, the human might easily judge that the perturbed inputs created using traditional methods, including mean, blur, and random noise replacement, are not aligned with the distribution of the original data set. In other words, the classifier did not observe such images during the training stage. Therefore, the behaviour of a classifier on these OOD inputs might not faithfully reflect what it has learnt from the training data set, and thus the provided explanations might be meaningless.

Recently, efforts have been made to develop *meaningful perturbation* methods aimed at mitigating the OOD problem in traditional perturbation-based methods. This is achieved by replacing the features to be perturbed with plausible alternative values that are inferred from the distribution of the training data set, as shown in Fig. 2.4. With plausible alternative values for certain features, their importance is evaluated through the following marginalisation form (Kim et al., 2020; Chang et al., 2019):

$$\begin{aligned} f(c|\boldsymbol{x}_{\setminus r}) &= \int f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r}) d\boldsymbol{x}_r^* \\ &= \mathbb{E}_{\boldsymbol{x}_r^* \sim p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r})}[f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r})] \end{aligned} \tag{2.5}$$



(a)

(b)

Figure 2.4: (a) Time series perturbation inputs created by replacing features with zeros and plausible values that inferred from the training data set; (b) The perturbation inputs for an image classification problem, where the centre region is replaced by mean, blur, random noises and plausible values inferred from the training data set. (This image is taken from (Chang et al., 2019)).

where r denotes a subset of the input features, $\boldsymbol{x}_r$ denotes a partition of the input $\boldsymbol{x} = \boldsymbol{x}_r \cup \boldsymbol{x}_{\setminus r}$, and $f(c|\boldsymbol{x}_{\setminus r})$ denotes the classifier's output when features $\boldsymbol{x}_r$ are perturbed by replacing them with plausible alternative values. The $f(c|\boldsymbol{x}_{\setminus r})$ can also be interpreted as the output of the classifier when $\boldsymbol{x}_r$ are unknown (or removed from the input) (Chang et al., 2019), since the contributions of $\boldsymbol{x}_r$ are marginalised (Kim et al., 2020; Chang et al., 2019). The difference between $f(c|\boldsymbol{x}_{\setminus r})$ and $f(c|\boldsymbol{x})$ measures the importance of $\boldsymbol{x}_r$. The $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$ in (2.5) describes the distribution of plausible values of $\boldsymbol{x}_r$ conditioned by $\boldsymbol{x}_{\setminus r}$, which guarantees that the perturbed inputs are aligned with the training data sets. However, the exact distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$ is difficult to achieve. Current promising approaches use generative models to approximate the distribution of the training data set (Kim et al., 2020; Chang et al., 2019) and then the trained generative model is used to generate within-distribution perturbed inputs for the specific instance on which the model's behaviour is going to be explained. Meaningful perturbation-

based methods have been used to explain image models (Chang et al., 2019) and language processing models (Kim et al., 2020), but in the context of MTSC, their performance has not been explored.

2.4.3 Proxy Explanation Methods

Unlike gradient-based and perturbation methods, which directly assess the importance of features by probing the model to be explained, proxy models adopt an indirect approach. They work by approximating the behaviour of the original model with a simpler, more interpretable model that can provide insights into the decision-making process. This approach has distinct advantages. For example, proxy models can offer interpretability for a wide range of models, including those where gradients are not available or where perturbation is computationally expensive or infeasible. Because proxy models simplify the behaviour of the original model, they can make the interpretation process more accessible to nonexperts, facilitating a better understanding of model decisions without the need for deep technical knowledge of the underlying AI technology.

Among proxy explanation methods, Local Interpretable Model-agnostic Explanations (LIME) stands out (Ribeiro et al., 2016) and is going to be described in detail. LIME initiates the explanation process by generating a proxy data set composed of the neighbours of the target input that needs to be explained. Then, an interpretable model is fitted to approximate the behaviours of the black-box model on this proxy data set. Subsequently, the explanation yielded by this interpretable model serves as a tool for understanding the behaviour of the black-box model on the particular target input. The specific implementations of LIME are as follows:

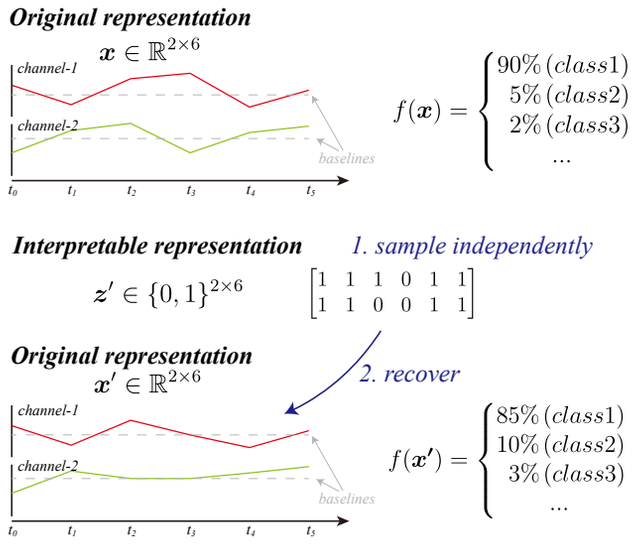


Figure 2.5: The traditional neighbour generation process for an MTSC problem. The input time series has 2 variables, each of which has 6 time steps. The baseline is established as the average value of the corresponding variable. The difference in class 1 probability between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ quantifies the contribution of three features marked as '0' in $\boldsymbol{z}'$ to the classifier's prediction that $\boldsymbol{x}$ belongs to class 1.

(1) *Creating neighbours:* This involves two steps as shown in Fig. 2.5:

- Sampling interpretable neighbours in the interpretable feature space, where each feature is denoted by 0 or 1,
- Recovering the sampled interpretable neighbours into the original feature space, where each feature is represented by a real value.

The interpretable neighbour, denoted $\boldsymbol{z}' \in \{0,1\}^{d\times T}$, maintains the same size as the original time series $\boldsymbol{x} \in \mathbb{R}^{d\times T}$, but with binary features. In the existing literature, the features of $\boldsymbol{z}'$ are sampled independently from a Bernoulli distribution with a probability of 0.5, which means that each feature has an equal probability of being '1' or '0'. The next step is to recover the obtained $\boldsymbol{z}'$ in the original feature space, denoted as $\boldsymbol{x}' \in \mathbb{R}^{d\times T}$. Features with a value of '1' in $\boldsymbol{z}'$ retain their original values, whereas those with a value of '0' are replaced with some baseline values, demonstrating that their contri-

butions to the model output are removed. In the context of time series, the baseline is typically established as the average value of the corresponding variable, as shown in Fig. 2.5.

(2) Assigning weights to neighbours: Within LIME, each neighbour is assigned a weight corresponding to its distance from the target. Neighbours closer to the target are assigned higher weights, demonstrating their greater importance in providing local explanations. Currently, in the literature, the weight for a neighbour, $\boldsymbol{x}'$, is determined by an exponential kernel:

$$\pi_x(\boldsymbol{z}') = exp(-D(\boldsymbol{x}', \boldsymbol{x}^t)/\sigma^2) \tag{2.6}$$

where D measures the distance between this neighbour and the target, $\boldsymbol{x}^t$. For sequence data, such as time series, distance measurement can be achieved by cosine similarity (Dong et al., 2006) or dynamic time warping (Berndt and Clifford, 1994). σ is an user-defined hyperparameter. Currently, in some widely used packages, the default setting of σ is $0.75 * \sqrt{N}$, where N represents the total number of features (Ribeiro et al., 2016).

(3) Creating a proxy data set: In the previous two steps, $\boldsymbol{x}'$, $\boldsymbol{z}'$ and $\pi_x(\boldsymbol{z}')$ are obtained. In this step, the predictions of the model on the created neighbours are evaluated, where y represents the probability that $\boldsymbol{x}'$ belongs to the target class that we are interested in. In LIME, a certain number of neighbours are generally created independently and compose a weighted proxy data set, denoted $\mathcal{D}(\boldsymbol{z}', y, \pi_x(\boldsymbol{z}'))$.

(4) Fitting an interpretable model on $\mathcal{D}(z', y, \pi_x(z'))$: The final step in LIME involves fitting a simple model to the proxy data set. The input of the interpretable model is $\boldsymbol{z}'$, while the output is y. $\pi_x(\boldsymbol{z}')$ denotes

the corresponding sample weight. If a linear model is adopted to approximate the original complex model, its coefficients can represent the importance of the corresponding features.

The work of (Lundberg and Lee, 2017) shows that the LIME framework can be adapted to compute SHAP values using the kernelSHAP method. Extending this approach, (Bento et al., 2021b) has adapted KernelSHAP for time series data, introducing an approach named TimeSHAP. KernelSHAP is grounded in game theory, which provides a robust theoretical framework to attribute the output of a model to its input features. To compute SHAP values, it is necessary to simulate the absence of certain characteristics by substituting them with baseline values. In the approach described by (Bento et al., 2021b), these baseline values are set as the mean values of the features being replaced. Therefore, this hard replacement inevitably disrupts the temporal dependency between features, resulting in the OOD problem.

2.4.4 Challenges of Feature Importance Explanations in MTSC

Although many feature importance explanation methods have been developed, their application within the context of MTSC presents distinct challenges that require careful consideration.

Challenges in identifying the most important features: Classic feature importance explanations aim to assign an importance score to each feature. However, determining accurate importance scores is challenging

and is considered a NP-hard problem.

$$\begin{aligned} \text{Binary Input} &: x_1, x_2, x_3 \\ \text{Binary Classifier} &: \mathrm{F}(x_1, x_2, x_3) = x_1 \wedge x_2 \wedge x_3 \\ \text{Current Input} &: \mathrm{F}(0, 0, 1) = 0 \end{aligned} \tag{2.7}$$

For example, (2.7) shows a classifier of which the three input features are binary. The classifier produces "1" when all input features are "1", otherwise it produces "0". Here, a classification task, $\mathrm{F}(0, 0, 1) = 0$, is explained using perturbation-based methods. If x_1 and x_2 are perturbed independently, which means that only one feature is perturbed each time, the classifier will maintain its prediction, since the classifier is saturated to produce "0" as long as one of the input features is "0". In this circumstance, both the importance of x_1 and x_2 is evaluated as zero. However, if x_1 and x_2 are perturbed together by replacing them with "1", the classifier will change its prediction. This means that x_1 and x_2 indeed contribute to the current prediction, but their contributions can only be evaluated together. This suggests that to obtain the importance score of one feature, we not only need to evaluate its independent contribution by perturbing this feature alone, but we also need to evaluate its cooperative contributions with other features by perturbing feature sets containing this feature. Therefore, achieving accurate feature importance is an NP-hard problem.

Optimisation-based feature importance explanation methods seek to overcome the challenges faced in obtaining accurate feature importance. Instead of assigning a score to each feature, these methods focus on identifying a critical group of features. This approach is guided by the question: *what is the smallest set of features of the current input that would maximally change the classifier's output?* These methods focus on identifying a group of features rather than assigning an importance score to each feature

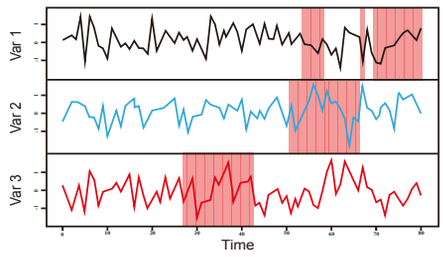


Figure 2.6: An optimisation-based feature importance explanation. Classic feature importance explanations (shown in Fig. 2.3) focus on assigning each feature an importance score, whereas optimisation-based methods identify a group of the most important features rather than exactly estimating the importance score of each feature.

(Fong and Vedaldi, 2017; Chang et al., 2019). An example of this kind of explanation is shown in Fig. 2.6. Compared to traditional feature importance explanation methods, optimisation-based approaches offer distinct advantages. For instance, they help humans concentrate on key features, facilitating a deeper exploration of the reasons behind a particular output. In contrast, traditional feature importance explanations often leave users without clear guidance on the number of features they should consider to fully understand an output.

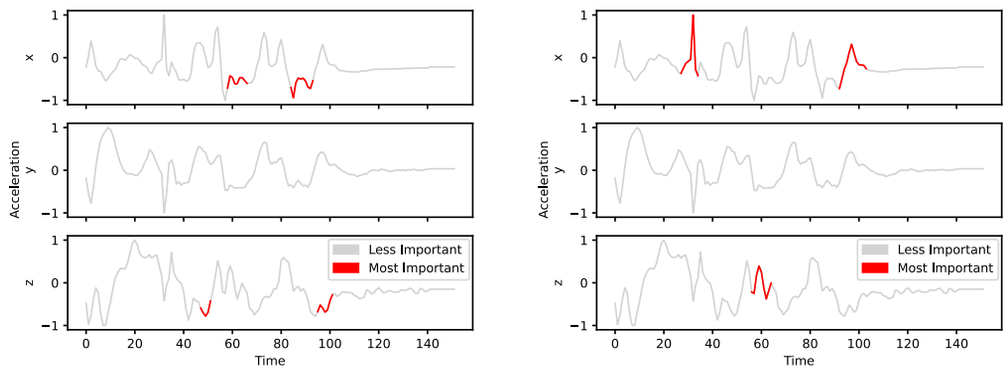
Although they have advantages, optimisation-based feature importance explanation methods face several challenges. The first comes from the difficulty in identifying the smallest feature set, which can only be achieved by exhaustive search. Current solutions mitigate this challenge through two types of approach: optimising an auxiliary neural network selector (Crabbé and Van Der Schaar, 2021; Chang et al., 2019) and using heuristic search methods (Ribeiro et al., 2018; Du and Xu, 2021; Vafa et al., 2021). The former approaches adopt gradient descent optimisation methods to train an auxiliary neural network to identify feature sets that can maximally change the classifier's output. A sparsity penalty is added to the loss function to obtain a small set. However, these approaches need the gradients of the classifiers, so they are not model-agnostic and cannot be applied to classi-

fiers whose gradients are not accessible. The approaches in the other group are model-agnostic and can be applied to any classifier. Typical heuristic search methods adopted in importance computation include Beam Search (Ribeiro et al., 2018), Greedy method (Vafa et al., 2021) and Genetic Algorithm (Du and Xu, 2021). Currently, these methods are adopted to explain language models (Vafa et al., 2021; Du and Xu, 2021), while in the context of MTSC, these methods have not been explored. The reason for this might be that MTS are usually high-dimensional, where these search methods face challenges in obtaining satisfying solutions.

Another challenge that optimisation-based feature importance methods face is the OOD problem. The basis of optimisation-based methods is to judge whether a group of features can maximally change the classifier's output. Currently, this is achieved by perturbing these features and evaluating the changes in the output. Therefore, optimisation-based feature importance methods also need to address the OOD problem. However, current methods do not consider the OOD problem (Crabbé and Van Der Schaar, 2021; Vafa et al., 2021; Ribeiro et al., 2018).

Stability of explanations: For feature importance explanation, it is crucial that the explanations provided are stable. Stability in this context means that the explanations for a given instance, as generated by the same method, should be consistent[3]. If the explanations provided by the same method differ when the method is performed multiple times, as shown in Fig. 2.7, they are less useful, as model users are unsure on which explanation to rely on (Zhou et al., 2021; Slack et al., 2021). Moreover, unstable explanations compromise their validity; a single feature should consistently yield the same importance score when evaluated under identical conditions

[3]The term 'stability' can carry different connotations across various works. For example, in some work (Situ et al., 2021; Slack et al., 2021), it is mentioned that the explanations for similar instances should also be similar.



(a) (b)

Figure 2.7: A multivariate time series from the Handwriting data set (Bagnall et al., 2018), where the acceleration of the hand in the x, y, and z directions is recorded by a smartwatch while a person is writing words. The trained classifier predicts that the person is writing the character 'E'. An explanation provided by LIME for this prediction is shown in (a), where the identified most important features are highlighted in red. However, when the same explanation process is carried out again, another explanation, different from (a), is obtained, and shown in (b).

by the same method.

Currently, various studies have highlighted the stability issue of LIME and some solutions have been proposed (Zhou et al., 2021; Slack et al., 2021; Visani et al., 2020). These focus on different steps of LIME, which are summarised in Table 2.1. Within LIME, increasing the number of neighbours will generally improve the stability of the explanations. With more neighbours, the decision boundary of the complex model can be captured more accurately and the randomness of the output surface captured by the neighbours decreases. However, more neighbours will make the explanation process very time consuming. To address this challenge, Slime (Zhou et al., 2021) adopts a hypothesis testing framework to determine whether the current number of neighbours is sufficient to obtain stable explanations. If the test fails, additional neighbours are created and added to the proxy data set until the test is passed. However, Slime adopts a completely random sampling method in sampling interpretable neighbours, which is not very efficient. In BayesLIME (Slack et al., 2021), the authors introduce

Table 2.1: Related works on improving stability of LIME

Method	Neighbours Generation		Weighting method
	Sample interpretable neighbours	Recover into real neighbours	
LIME (Ribeiro et al., 2016)	iid ∼Bern (0.5)	Traditional**	(2.6) with fixed σ***
Slime* (Zhou et al., 2021)	iid ∼Bern (0.5)	Traditional**	(2.6) with fixed σ***
BayesLIME (Slack et al., 2021)	Fosused sampling	Traditional**	(2.6) with fixed σ***
OptiLIME (Visani et al., 2020)	iid ∼Bern (0.5)	Traditional**	(2.6) with optimised σ

*Increase neighbours based on a hypothesis testing framework
**Traditional: replace features with baseline values.
***σ is set to $0.75 * \sqrt{N}$ (N is the total number of features).

a Bayesian framework to estimate the uncertainty of feature importance provided by LIME and to determine the number of neighbours needed for stable explanations. Based on the proposed framework, they propose a focused sampling strategy to sample interpretable neighbours, which has been shown to be more efficient. OptiLIME (Visani et al., 2020) emphasises the significant influence of σ in (2.6) on the stability of the explanations. It proposes adopting optimisation methods to find the optimal σ.

However, it is important to note that all of these methods utilise traditional methods to create neighbours, for example, replacing features whose contributions are going to be removed, with specific baselines, which have the risk of resulting in OOD inputs. The OOD problem might be one of the key reasons for unstable explanations. Investigating and resolving the influence of the OOD problem on the stability of the LIME framework is necessary.

2.5 Counterfactual Explanations

While feature importance metrics can highlight the inputs that significantly influence a model's decisions, these kinds of explanation may not always effectively convey the model's behaviour in a manner that is easily understood by the layperson. For such individuals, the primary concern often revolves around understanding how specific changes to input could alter the model's output to achieve a desired result so as to do some decision-making tasks. In this context, counterfactual explanations emerge as a powerful tool. Counterfactual explanations aim to answer the following question. How can altering the input change the classifier's prediction to a different (or desired) outcome? By providing this kind of information, counterfactual explanations offer users a more understandable explanation and help them think of possible actions to achieve desired outcomes. In this section, the properties that a good counterfactual explanation should process are first described. Then, the approaches developed to provide counterfactual explanations are analysed.

2.5.1 Properties of counterfactual explanations

Research in counterfactual explanations aims to derive explanations adhering to certain desirable properties, such as proximity, sparsity, and plausibility. These properties aim to provide users with more intuitive, easily comprehensible, and relevant insights on the decision-making process of complex models.

Proximity: The counterfactual $\boldsymbol{x}'$ should be similar to the original input $\boldsymbol{x}$, i.e., given a distance measurement, the distance between $\boldsymbol{x}'$ and $\boldsymbol{x}$ should be as small as possible. Proximity is often referred to as similarity.

Sparsity: Sparsity refers to the principle of making the smallest number of changes necessary to the features of the input to alter the outcome of the classifier. A more sparse counterfactual will have fewer differences compared to the original input. This is helpful in providing comprehensible explanations, because it lets users focus on the most important changes.

Plausibility: The plausibility of a counterfactual explanation measures how likely it is to have come from the same data distribution as the training set. This aspect is vital because it ensures that the counterfactuals are realistic and practically possible. If a counterfactual is implausible, it might suggest a scenario that is very unlikely or even impossible in real life. Such explanations can be misleading and not very useful in practical situations. Plausibility is also often referred to as feasibility or reliability. To assess plausibility, various methods have been proposed. For example, Laugel et al. suggest evaluating the plausibility of a counterfactual by using a concept known as the ϵ-chain distance (Laugel et al., 2019). Another approach, suggested by Hammer et al., argues that a plausible counterfactual should be located in a densely populated area of the data space (Artelt and Hammer, 2020).

Diversity: The diversity property of counterfactual explanations plays a crucial role in enhancing their effectiveness and applicability. Diversity ensures that a range of distinct, yet plausible scenarios are presented, each illustrating a different path by which the outcome of a model could be altered. This variety is vital because it acknowledges the multifaceted nature of real-world situations, where multiple factors can lead to the same result. By providing a spectrum of various counterfactuals, these explanations enable users to understand not just one possible change, but several potential variations and their respective impacts on the outcome. This in turn facilitates a more comprehensive understanding of the model's behaviour,

fostering greater trust and transparency. Additionally, the diverse nature of these counterfactuals is particularly important in complex decision environments, such as healthcare or finance, where understanding the range of influencing factors can significantly aid decision-making processes, ensuring more robust, informed, and equitable outcomes.

Stability: Stability stands as a critical property in the realm of counterfactual explanations. Despite its importance, it should be noted that only a limited number of studies have focused on this aspect. Stability refers to the ability to consistently generate the same explanations for a given instance using identical methods and parameter settings, ensuring that the explanations are reliable and free from the influence of some randomness components. For example, some counterfactual explanation methods generate explanations from a randomly sampled input. This randomness results in variations in the final results. Inconsistent results from the same methodology could not only lead to confusion but also undermine the credibility of the explanations. In practical terms, this means that when a user applies a counterfactual method to understand a model's decision, they expect to receive a consistent explanation each time, provided that the input and conditions remain unchanged. This consistency is particularly vital in safe critical applications, such as healthcare or finance, where varying explanations could lead to significantly different interpretations and actions.

Diversity and stability seem to be in conflict, but they address different properties of counterfactual explanations. Diversity in this context refers to the range of potential actions or changes suggested by the explanations for a specific instance. It focusses on uncovering a variety of plausible counterfactuals that could alter the model's outcome, thereby providing a comprehensive understanding of the different factors influencing the decision. However, stability emphasises the reliability of these potential ac-

tions. It ensures that the set of suggested changes, or the counterfactuals, remains consistent and unaffected by randomness components existing in the explaining process when the same instance is evaluated under the same conditions with the same method. In simpler terms, diversity seeks to generate a range of different explanations, for example, offering 10 distinct counterfactuals for a given scenario. On the contrary, stability ensures that these 10 explanations remain consistent and reliable; when the explanation process is repeated under the same conditions, it consistently yields the same set of 10 counterfactuals.

2.5.2 Approaches to provide counterfactual explanations in MTSC

Currently, some counterfactual explanation methods have been proposed. Among these methods, the WACH method, introduced by Wachter et al. (Wachter et al., 2017), is the most famous one. Although not originally designed to address MTSC problems, it has been applied in this context (Delaney et al., 2021b). WACH generates counterfactual explanations by optimising the following objective.

$$\underset{\boldsymbol{x}'}{\operatorname{argmin}} \mathcal{L} = (f(c|\boldsymbol{x}') - c')^2 + \lambda d(\boldsymbol{x}, \boldsymbol{x}') \tag{2.8}$$

where the c' is the desired outcome of the classifier, often defined as the predicted probability of a specific class; $d(\boldsymbol{x}, \boldsymbol{x}')$ measures the distance between the counterfactual $\boldsymbol{x}'$ and the original input $\boldsymbol{x}$, serving as a measure of similarity between these two time series; λ serves as a crucial hyperparameter that balances the objective of aligning the model's output closely with the desired outcome and maintaining a high degree of similarity between the

counterfactual and the original input.

In the literature, some counterfactual explanation methods tailored for MTSC have been developed (Delaney et al., 2021b; Ates et al., 2021b; Höllig et al., 2022; Bahri et al., 2022). Among these, the *Naive Guide* (Delaney et al., 2021b) uses a two-step process. In the first step, it employs feature importance techniques, such as Class Activation Mapping (CAM) (Selvaraju et al., 2017b) or Shapley Values (Castro et al., 2009), to identify the most distinctive segments of the input time series that significantly influence the classifier's decision. Subsequently, these identified segments are replaced with alternative candidates from the data set with the goal of generating a counterfactual instance that alters the original classification outcome. The studies (Ates et al., 2021b; Höllig et al., 2022) propose to adopt search algorithms for counterfactual generation. These algorithms focus on searching for the optimal segment from a candidate data set that can be used to replace specific parts of the input time series. The search process is driven by multiple objectives, such as altering the outcome of the classifier to the desired one, maximising the similarity between the counterfactual and the original input, and enhancing the plausibility of the obtained counterfactuals. The latter objective aims to obtain realistic counterfactual explanations. In the study (Bahri et al., 2022), a novel shapelet-based counterfactual generation method is introduced. This approach tries to identify and replace the most distinctive shapelet in the original time series with a corresponding shapelet associated with the desired label. This method takes advantage of the inherent characteristics of shapelets, key subsequences that significantly influence the classifier's decision, to effectively steer the model's prediction towards a specific outcome.

However, a common limitation of these approaches is that they directly replace the segment using candidates from the data set. Such direct replace-

ment might potentially disrupt the temporal and feature dependencies that are intrinsic in time series data, which might result in unrealistic counterfactuals that may not fully align with the distribution of the real data set on which the classifier is trained. In this scenario, the altered prediction of the model is not attributed to meaningful changes, but rather to issues stemming from the OOD problem. This raises concerns about the realism and practical applicability of the counterfactuals obtained. Addressing this challenge requires careful consideration of the distributions of the original time series data, ensuring that generated counterfactuals are not only effective in changing the classifier decision but also align with the real-world contexts.

In the MTSC realm, some methods have been developed to offer counterfactual explanations that consider the distribution of training data. For example, Meng et al. (Meng et al., 2023) design a method based on greedy segmentation and identification to pinpoint critical segments of the input time series and then employ a generative model that is trained to approximate the data distribution to create plausible alternative segments. This approach is designed to minimise the likelihood of producing unrealistic counterfactuals. However, the efficiency of this framework is highly dependent on the capacity of the generative model. If the generative model adopted is not sufficiently accurate to approximate the data distribution, it may still lead to the generation of unrealistic counterfactuals.

In addition to the methods mentioned above for counterfactual explanations, the Generative Adversarial Network (GAN) framework (Arjovsky et al., 2017) has been adapted to address MTSC problems (Lang et al., 2023a). The fundamental concept of GAN involves two key components: a discriminator and a generator. These components work collaboratively to produce counterfactuals that are consistent with the data distribution

of the original data set. The generator creates potential counterfactuals, while the discriminator evaluates their authenticity, ensuring that the generated counterfactuals are indistinguishable from real data instances. However, applying the GAN framework to counterfactual generation presents challenges. For example, GANs are known for their training difficulties, primarily due to the adversarial setup where the generator and discriminator compete. This competition can lead to instability, particularly when dealing with the increased complexity of time series data.

Chapter 3

Feature Importance Explanation Framework through Meaningful Perturbation and Optimisation

3.1 Introduction

To promote the development of XAI in the context of MTSC, a foundational step is to consider the types of explanations to be provided. For a specific audience, including data scientists and model developers, pinpointing the sequences (or subsequences) within the time series that significantly impact the model's predictions is of paramount importance. Such insights are crucial for enhancing the model's performance through approaches like feature selection and model simplification. In this context, feature importance explanations become particularly relevant. These explanations shed light on which parts of the input time series are most influential to the

model's predictions, guiding users to fine-tune their models.

The previous chapter highlighted several challenges encountered by current start-of-the-art feature importance explanation methods. One of the challenges involves addressing the OOD issues inherent in traditional perturbation methods (Hase et al., 2021). Traditional perturbation methods are usually implemented by replacing the features under evaluation with zeros or adding random noise. The resulting changes in the classifier's output indicate the importance of these features. However, such perturbation approaches might create unrealistic inputs that the classifier has never encountered during the training stage. Consequently, the predictions made on these inputs may not be reliable and do not accurately reflect the importance of the perturbed features. This issue is well recognised in image (Chang et al., 2019) and language models (Kim et al., 2020). Due to the significant temporal dependency inherent in time series data, such traditional perturbation operations can easily disrupt the temporal dependency of time series data and result in unrealistic inputs. However, this problem has not received sufficient attention in MTSC.

Another challenge encountered by feature importance explanation methods relates to the joint contributions of features. As discussed in Section 2.4.4, assessing the importance of certain features can only be done in conjunction with others, making the precise evaluation of feature importance an NP-hard problem. To address this, optimisation-based approaches attempt to answer: What is the smallest set of features in the current input that, if altered, would significantly change the classifier's output (or alter the current predicted label)? These methods do not concentrate on precisely evaluating the importance score of each feature. Instead, they aim to identify the minimal set of features whose modification in current values can alter the classifier's output. Therefore, currently in the literature, some op-

timisation search methods (Ribeiro et al., 2018; Vafa et al., 2021; Du and Xu, 2021) have been adopted to find such an important group of features. However, these optimisation approaches have not yet been developed in the MTSC context.

In this chapter, we propose a comprehensive feature importance explanation framework. Specifically, this framework employs generative models to address the OOD issues inherent in traditional perturbation methods. It also integrates an optimisation approach specifically designed for MTSC, with the aim of identifying the most important features.

The rest of this chapter is organised as follows. The problem we are going to address is formulated in Section 3.2. An overview of our proposed framework is provided in Section 3.3. After that, the framework, including the design of the generative model (Section 3.3.1) and the design of the search method (Section 3.3.2), which comprise this framework, are described in detail. Following these, the experimental design is provided in Section 3.4 and the corresponding results and analysis are provided in Section 3.5.

3.2 Problem definition

Let $f(c|\boldsymbol{x})$ be a MTSC that produces the probability of $\boldsymbol{x}$ belonging to class c. $f(\boldsymbol{x})$ produces the label assigned to $\boldsymbol{x}$, which is the one with the highest probability. Assume that we have a particular input $\boldsymbol{x}^*$ for which the classifier predicts its label should be c_1. Our main goal is to explain, using a feature importance explanation, why the classifier predicts that $\boldsymbol{x}^*$ belongs to class c_1 rather than another class. In the final explanation, the key features of $\boldsymbol{x}^*$ that are necessary for the classifier to produce this output are identified. If these features had not taken their current values, but

other plausible values instead, the classifier would have produced a different output. In other words, these features support the classifier in making the current prediction. Therefore, these key features are also called 'supporting features' and denoted by $\boldsymbol{x}_s^*$. The set $\boldsymbol{x}^*$, which includes all input features, is definitely a set of supporting features. However, a smaller set would provide more insight into explaining the output. Therefore, to provide better explanations, the proposed framework tries to find the smallest set of supporting features to help users understand why $f(\boldsymbol{x}^*) = c_1$.

3.3 Pipeline of the proposed framework

Fig. 3.1 shows the pipeline of the proposed framework. Algorithm 1 provides the corresponding pseudo-code for this framework. Since this framework takes a post-hoc approach, a classifier p is first trained on the data set. Then an output is produced for a given instance, $\boldsymbol{x}$. To explain this output, a generative model G is trained to learn the distribution of the data set on which the classifier is trained and then used to generate perturbed inputs for the input to be explained. Since the training data set is fixed, the generative model only needs to be trained once and can be used to explain any input coming from the same data set. The structure of the generative model and the training strategy are described in Section 3.3.1. After that, a greedy-based segmentation and identification search method is designed to identify the smallest supporting feature set of this input. This method consists of segments and identification steps. A detailed description of these two steps is provided in Section 3.3.2. One thing that has to be noted is the connection between the proposed generative model and the search method. In this framework, the generative model is primarily used to create realistic inputs for evaluating the importance of specific features, while the greedy

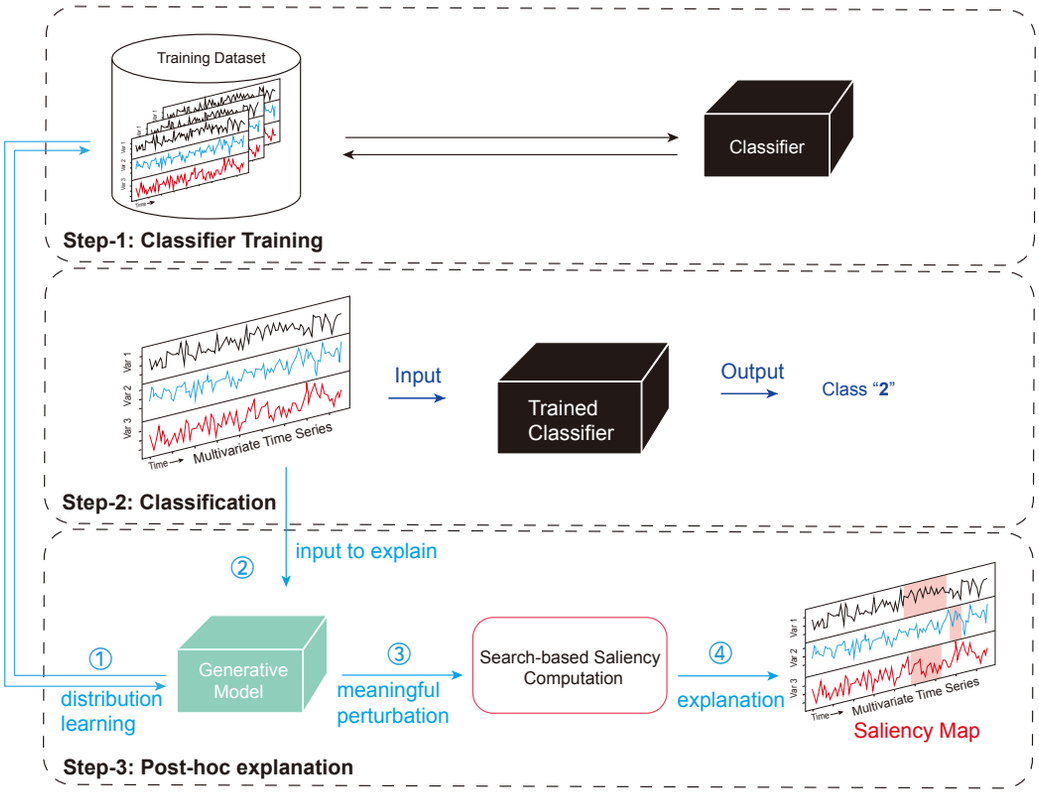


Figure 3.1: Explaining a classifier through a post-hoc approach. In the first step, the classifier is trained on an MTSC training data set. Then, the trained classifier is used to classify an instance of the MTSC, of which the output is explained in the third step. In the third step, the generative model is optimised to learn the distribution of the training data set (as indicated by arrow 1). Subsequently, the trained generative model generates meaningful perturbation samples for the input that needs explanation (as indicated by arrow 2). These samples are then utilized in the designed search methods (as indicated by arrow 3). Finally, the saliency map is produced using the designated search-based saliency computation method (as indicated by arrow 4).

segmentation strategy is used to identify the most important feature set for a given output. During the greedy search process, the generative model is used to generate realistic inputs. Finally, an explanation is generated for this output through a saliency map, where a set of supporting feature sets is highlighted.

Algorithm 1 The pseudo-code for the proposed framework

Inputs:
  $\boldsymbol{x} \leftarrow$ Input Time Series;
  $G \leftarrow$ Training the generative model;
  Segments $\leftarrow$ Segmentation($\boldsymbol{x}$);
while True **do**
    $\boldsymbol{W}_0 \leftarrow$ Initializing saliency maps with Segments;
    $\boldsymbol{W}_{\text{best}} \leftarrow$ Searching for the best saliency map using $(\boldsymbol{x}, G, \boldsymbol{W}_0, p)$;
    Identified Segments $\leftarrow \boldsymbol{W}_{\text{best}}$;
    if the length of Identified Segments $= 1$ **then**
      break;
    end if
    Segments $\leftarrow$ Segmentation(Identified Segments);
end while
  return $\boldsymbol{W}_{\text{best}}$

3.3.1 Designing the generative model

The basis of this framework is to determine whether a group of features is supporting, which is achieved by (2.5). If $f(\boldsymbol{x}^*_{\setminus r}) \neq f(\boldsymbol{x}^*)$, $\boldsymbol{x}^*_r$ are supporting features. Here, a generative model for MTS is designed to estimate the necessary distribution, $p(\boldsymbol{x}^*_r|\boldsymbol{x}^*_{\setminus r})$. The structure of the proposed generative model is shown in Fig. 3.2. In the remainder of this section, the method for generating training data for this generative model is described first. Then, a detailed description of the structure of the generative model is provided. Finally, the objectives to be optimised and the training strategy are given.

Training data generation. To train the generative model to estimate the distribution $p(\boldsymbol{x}^*_r|\boldsymbol{x}^*_{\setminus r})$, the first step is to obtain its inputs, $\boldsymbol{x}^*_{\setminus r}$, and outputs, $\boldsymbol{x}^*_r$. We sample MTS data from the data set and for each MTS, a binary mask is created to mask out certain features through element-wise multiplication, $\boldsymbol{x}^* \odot \boldsymbol{M}$, creating an input for the generative model. Then, the complementary components $\boldsymbol{x}^* \odot (\boldsymbol{1} - \boldsymbol{M})$ are the target that the generative model tries to generate. The purpose of these training procedures is to enable the model to predict plausible values for $\boldsymbol{x}^*_r$ (represented by

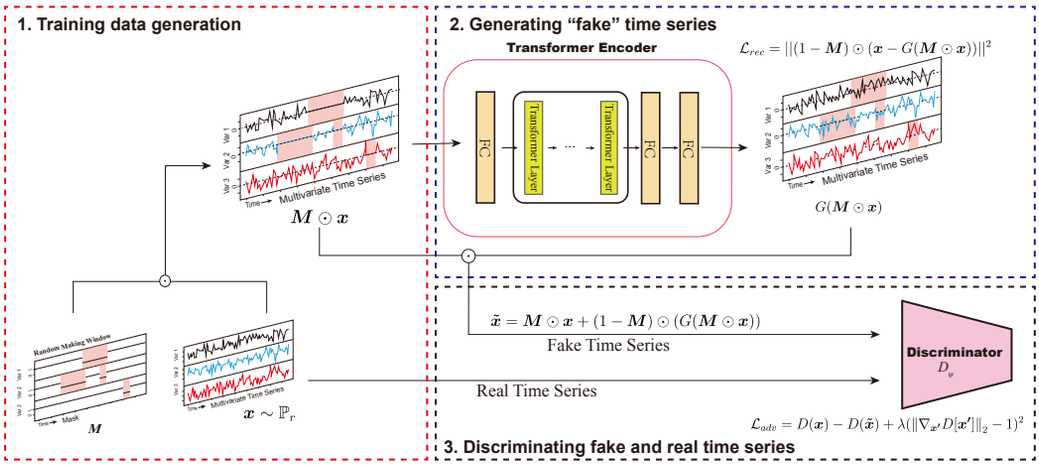


Figure 3.2: The structure of the proposed generative model. The training data are generated by masking features within some randomly chosen time intervals. Then, the generative model is trained to generate plausible alternative values for these masked features. To enhance the quality of the MTS generated by the generator, a discriminator is added to distinguish the fake MTS (generated by the generative model) and the real MTS (sampled from the data set).

$\boldsymbol{x}^* \odot (\mathbf{1} - \boldsymbol{M})$) when conditioned on $\boldsymbol{x}^*_{\setminus r}$ (represented by $\boldsymbol{x}^* \odot \boldsymbol{M}$).

Generative model architecture. The objective of the generative model is to create plausible values for $\boldsymbol{x}^*_r$ conditioned by $\boldsymbol{x}^*_{\setminus r}$. This is very similar to time series imputation problems, where the missing components of a time series need to be filled in. However, in the missing value imputation problem, they do not have the actual ground-truth values. Therefore, imputation models attempt to learn the temporal dependence of the time series using the observed values. The missing parts are then filled in on the basis of the learnt temporal dependence. In our scenario, the time series we have are complete. We have the ground truth for each feature, so the model can be trained to predict the target features. Our goal is to generate alternative plausible values for certain features that can maintain the perturbed time series within the original distribution. Nevertheless, although time series missing imputation models have a different goal, they can be applied in this situation. Before the development of the generative model presented here, the performance of various advanced time series

missing value imputation methods (Cao et al., 2018) with the proposed Transformer-based generative model (Vaswani et al., 2017) is compared. The performance of the imputation methods and the proposed method can be obtained from the supplementary material of our published work (Meng et al., 2023). The results indicate that the designed generative model outperforms existing imputation methods, justifying its adoption for the work in this framework.

The architecture of the designed generative model is shown in Fig. 3.2. Specifically, the original time series is first embedded by a fully connected layer and then fed into a Transformer encoder, which is composed of several Transformer layers. The initial fully connected layer is used to transform the input time series data into a higher-dimensional space to better capture complex patterns and dependencies. This is a common practice in deep learning for time series as it allows the model to start the transformation process with a richer representation of the input data. The choice of a Transformer encoder is motivated by its proven capability in handling sequential data, particularly due to its self-attention mechanism, which effectively captures temporal relationships across different time steps. The encoder's output then passes through another fully connected layer, which is coupled with a hyperbolic tangent (tanh) activation function. This design restricts the predicted values within the range [-1, 1], which is aligned with the data preprocessing method used in this study (described in Section 3.4).

Optimisation Objective. The generative model aims to produce the masked components. Therefore, one of the loss functions is a reconstruction loss:

$$\mathcal{L}_{rec} = ||(1 - \boldsymbol{M}) \odot (\boldsymbol{x} - G(\boldsymbol{M} \odot \boldsymbol{x}))||^2 \tag{3.1}$$

where G denotes the whole generative model. This loss measures the difference between the target features of the original inputs and those generated by the generative model. Additionally, to improve the quality of the MTS generated by the model, we adopt the adversarial training strategy. Specifically, we take the Wassersten Generative Adversarial Networks (WGANs) (Gulrajani et al., 2017) diagram to train the generative model. WGANs are a modification of classic GANs (Goodfellow et al., 2014) whose discriminator has no activation function in its final layer. WGANs are better at learning stability and avoiding model collapse than classic GANs (Arjovsky et al., 2017). In this paper, we use the gradient penalty proposed in (Gulrajani et al., 2017) to implement the Lipschitz constraint of the WGANs. The adversarial loss is

$$\mathcal{L}_{adv} = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] + \lambda \mathop{\mathbb{E}}_{\boldsymbol{x'}\sim\mathbb{P}_{x'}}[(\|\nabla_{\boldsymbol{x'}} D[\boldsymbol{x'}]\|_2 - 1)^2] \tag{3.2}$$

where D is the discriminator; P_r is the training data set distribution; P_g is the fake data set distribution from which the MTS are generated by $\tilde{\boldsymbol{x}} = \boldsymbol{M} \odot \boldsymbol{x} + (1 - \boldsymbol{M}) \odot (G(\boldsymbol{M} \odot \boldsymbol{x}))$, being the combination of the generated masked features and the non-masked parts of the real data; $P_{x'}$ is the distribution sampled uniformly along straight lines between pairs of points sampled from P_r and P_g (Gulrajani et al., 2017); λ is a parameter that controls the strength of the gradient penalty.

Training Strategy. Following the GAN training diagram (Goodfellow et al., 2014), the discriminator D and the generator G are optimised iteratively through stochastic gradient descent. The discriminator is optimised by minimising the adversarial loss $\mathcal{L}_{adv}$ in (3.2) and the generator is opti-

mised by minimising the overall generative loss which is expressed by:

$$\mathcal{L}_{rec} = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\mathcal{L}_{rec} + \lambda_{dis} D(\tilde{\boldsymbol{x}})] \tag{3.3}$$

where λ_{dis} is a hyperparameter controlling the strength of the penalty of the adversarial loss.

3.3.2 Searching for the most important features for MTSC

The previous steps have designed a generative model to estimate the distribution necessary to judge whether a set of features is supporting for the given classification. The next step of the proposed framework is to find the smallest set of supporting features. However, the smallest supporting feature set can only be obtained by exhaustive search, which is impossible when the feature space is huge. Therefore, we propose to use heuristic search-based methods to find approximate solutions in a reasonable time. This section first introduces the objective (or fitness) function that guides the search. Then, we design a method to address the difficulties faced by classic search methods when searching in a huge feature space.

Fitness function. Before introducing the fitness function, we will describe how to evaluate whether an explanation is ideal or not and then formulate these evaluation metrics as a fitness function. A lower value of the fitness function means that a better explanation is achieved.

First, the features identified in an ideal explanation should be supporting, which means that if these features were replaced by other plausible values, the classifier would produce a different output. This criteria is formulated

as follows:

$$\text{Sup}(\boldsymbol{W}) = \begin{cases} 0 & \text{if } f(c|\boldsymbol{x}) = f(c|\boldsymbol{x}_{\setminus r}) \\ 1 & \text{if } f(c|\boldsymbol{x}) \neq f(c|\boldsymbol{x}_{\setminus r}) \end{cases} \tag{3.4}$$
$$\text{where } \boldsymbol{x}_{i,j} \in \boldsymbol{x}_{\setminus r}, \text{ if } \boldsymbol{W}_{i,j} = 1$$

where the feature importance map is represented by a binary mask $\boldsymbol{W} \in \mathbb{R}^{D \times T}$; $\boldsymbol{W}_{d,t} = 1$ means the feature $\boldsymbol{x}_{d,t}$ is identified; $\text{Sup}(\boldsymbol{W}) = 1$ means the identified features are supporting features. To calculate $f(c|\boldsymbol{x}_{\setminus r})$ using (2.5), it is necessary to sample instances from the distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$. The work (Gal and Ghahramani, 2016) has shown that keeping the dropout layer active during the inference stage and running the forward process multiple times is equivalent to sampling samples from the learnt distribution. Therefore, (2.5) can be reformulated as:

$$\begin{aligned} f(c|\boldsymbol{x}_{\setminus r}) &= \int f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r}) d\boldsymbol{x}_r^* \\ &= \int f(c|g_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r}), \boldsymbol{x}_{\setminus r}) d\boldsymbol{\epsilon} \\ &= \mathbb{E}_{\hat{\boldsymbol{\epsilon}} \sim p(\boldsymbol{\epsilon})} f(c|G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r}), \boldsymbol{x}_{\setminus r}) \end{aligned} \tag{3.5}$$

where G represents the generative model; $p(\boldsymbol{\epsilon})$ is a product of Bernoulli distribution with probabilities of $1-p$, and p represents the dropout rate in the training stage; the $\boldsymbol{\theta}$ represent the weights of the final fitted generative model.

Second, sparsity is also a desirable property of an ideal explanation, which means that the size of the identified supporting feature set should be small. This helps in providing clear explanations. This objective is expressed by:

$$\text{Spar}(\boldsymbol{W}) = \|\boldsymbol{W}\|_1 \tag{3.6}$$

where $\|\cdot\|_1$ denotes the L1-norm of the binary mask $\boldsymbol{W}$. A lower value of $\text{Spar}(\boldsymbol{W})$ means that fewer features are identified.

Finally, similar to (Crabbé and Van Der Schaar, 2021), we assume that in the context of MTS, features within continuous time steps provide more insights to humans to understand the behaviour of a classifier than those scattered over discrete time steps. This is similar to the pixels in images, where the pixels within a continuous region make more sense for humans than those scattered everywhere. Therefore, in an ideal explanation, we hope the identified features can be continuous in time, which can be realised by minimising the following objective:

$$\text{Conti}(\boldsymbol{W}) = \frac{1}{DT} \sum_{d=0}^{D} \sum_{t=0}^{T-1} |\boldsymbol{W}_{d,t} - \boldsymbol{W}_{d,t+1}| \tag{3.7}$$

The final fitness function for the task at hand is the combination of the above three objectives:

$$\text{Fitness}(\boldsymbol{W}) = \lambda_e(1 - \text{Sup}(\boldsymbol{W})) + \lambda_s \text{Spar}(\boldsymbol{W}) + \lambda_c \text{Conti}(\boldsymbol{W}) \tag{3.8}$$

where λ_e, λ_s, and λ_c are three parameters that control the priority of these objectives in the process of searching for an ideal explanation. Finding supporting features has the highest priority, therefore, the fitness value of a saliency map, of which the identified features are supporting features, should always be smaller than that of which the identified features are not supporting features. This priority is achieved by setting $\lambda_e > \lambda_s \text{Spar}(\boldsymbol{W}) + \lambda_c \text{Conti}(\boldsymbol{W})$ for any $\boldsymbol{W}$. Similarly, the sparsity requirement has the second priority, which is maintained by setting $\lambda_s \text{Spar}(\boldsymbol{W}) > \lambda_c \text{Conti}(\boldsymbol{W})$ for any solution. Because $\text{Spar}(\boldsymbol{W}) \subset [0, DT]$ and $\text{Conti}(\boldsymbol{W}) \subset [0, 1]$, in this thesis, to maintain the defined priority, the following settings are chosen:

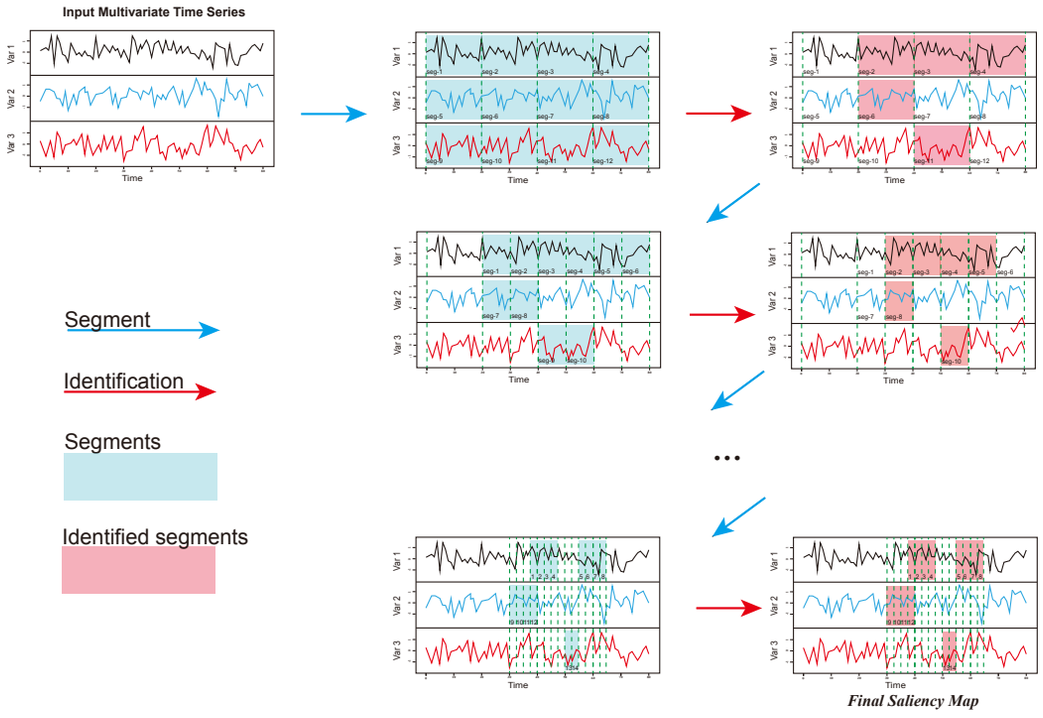


Figure 3.3: Proposed greedy-based segmentation and identification search method. The input MTS are segmented into several segments. In the next step, the smallest supporting segments are identified through a binary search method. In the next segment step, the previously identified segments are further segmented into smaller segments. Then, the adopted binary search model is implemented on these smaller segments. This process continues until the length of the segments is one.

$\lambda_e = 10DT$, $\lambda_s = 1.0$, and $\lambda_c = 0.1$.

Greedy-based segmentation and identification search method. After defining the fitness function, the challenge we are facing now is how to obtain an ideal explanation in a reasonable time. MTS are often high-dimensional, on which classic search methods face difficulties in obtaining ideal solutions. Here, we mitigate this difficulty by the proposed greedy-based segmentation and identification search method, of which an overview is shown in Fig. 3.3. The overall search operation is made up of a number of sequential substeps. Each substep contains two operations: segmentation and identification. At first, the original input MTS is segmented into a number of non-overlap segments. Each segment contains features within a certain time interval and is considered as a 'super-feature' during

the next identification step. In the identification steps, a binary heuristic search method is used to search for the smallest group of supporting segments over the feature space composed of these 'super-features'. Then in the next segment step, the identified supporting 'super-features' in the previous identification step are further segmented into smaller segments, on which the binary search method is carried out again. The two operations are carried out iteratively until the 'super-features' cannot be segmented further (the length of the identified segments is one).

In the identification steps, we are dealing with a optimisation problem, where many classic heuristic methods can be adopted, such as Genetic Algorithms (GA) (Mitchell, 1998) and Binary Particle Swarm Optimisation methods (BPSO) (Miranda, 2018). Both of these classic methods have their advantages and limitations. In this work, we implement both GA and BPSO in the identification steps. We will show in the experiment that these two methods achieve very similar performance. One thing that should be noted here is that although we take GA and BPSO, users can use any search method in the identification steps. We believe that a suitable search method will help to achieve better results. In addition, although we focus on the MTSC context, the proposed greedy-based segmentation and identification search strategy is more broadly applicable to any domain, such as images.

3.4 Experimental Design

In this section, details of the experimental design are provided. First, we describe the adopted MTSC data sets and the classifier that we are going to explain. Then, the experimental parameters, including the hyperparame-

ters of the generative model and the settings of the adopted search methods, are provided. Finally, the benchmarking explaining methods that are going to be compared with are presented, and the quantitative evaluation metric is described.

3.4.1 Data sets

To validate the performance of the proposed framework, the UEA MTSC archive is adopted, which is a well-known benchmarking archive containing 30 MTSC data sets collected from a wide range of applications (Bagnall et al., 2018). In addition, the handwriting image data set, MNIST (Lecun et al., 1998), is also used in our experiments. Although it was not originally collected as an MTS, we can treat it as an MTS, as some work has done (Ismail et al., 2020). The MNIST is taken here for visualisation purposes, which helps to obtain some useful qualitative evaluations. The details of the preprocessing methods used are described as follows.

UEA TSC Archive: Detailed information on these data sets can be obtained from work (Bagnall et al., 2018), and a summary of these 30 data sets is given in Table 3.1. As shown in the table, some datasets contain only two labels, while others have more than two labels. Therefore, the classifier to be optimised can be either a binary classifier or a multi-class classifier, depending on the number of labels. However, it is important to note that all these datasets are balanced, which means that within each dataset, every label has the same number of samples. As can be seen in the table, the time series in some data sets have too many features or time steps, e.g. DuckDuckGeese, EigenWorms, MotorImagery, and PEMS-SF. Providing explanations for these problems is very time-consuming. In this paper, we only consider 26 medium-sized data sets. The data sets, includ-

Table 3.1: A summary of the 30 data sets in the UEA TSC Archive

	data set	Training Cases	Test Cases	D	T	Classes
0	ArticularyWordRecognition	275	300	9	144	25
1	AtrialFibrillation	15	15	2	640	3
2	BasicMotions	40	40	6	100	4
3	CharacterTrajectories	1422	1436	3	182	20
4	Cricket	108	72	6	1197	12
5	**DuckDuckGeese***	50	50	1345	270	5
6	**EigenWorms***	128	131	6	17984	5
7	Epilepsy	137	138	3	206	4
8	EthanolConcentration	261	263	3	1751	4
9	ERing	30	270	4	65	6
10	FaceDetection	5890	3524	144	62	2
11	FingerMovements	316	100	28	50	2
12	HandMovementDirection	160	74	10	400	4
13	Handwriting	150	850	3	152	26
14	Heartbeat	204	205	61	405	2
15	InsectWingbeat	30000	25000	200	22	10
16	JapaneseVowels	270	370	12	29	9
17	Libras	180	180	2	45	15
18	LSST	2459	2466	6	36	14
19	**MotorImagery***	278	100	64	3000	2
20	NATOPS	180	180	24	51	6
21	PenDigits	7494	3498	2	8	10
22	**PEMS-SF***	267	173	963	144	7
23	PhonemeSpectra	3315	3353	11	217	39
24	RacketSports	151	152	6	30	4
25	SelfRegulationSCP1	268	293	6	896	2
26	SelfRegulationSCP2	200	180	7	1152	2
27	SpokenArabicDigits	6599	2199	13	93	10
28	StandWalkJump	12	15	4	2500	3
29	UWaveGestureLibrary	120	320	3	315	8

* these 4 data sets are not considered in this thesis. Providing explanations for these problems would be very time-consuming.

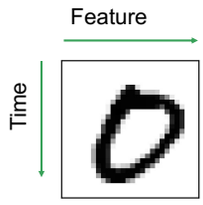


Figure 3.4: Treating MNIST images as MTS. Each image is regarded an MTS with 28 time steps, each of which has 28 features

ing CharacterTrajectories (30%), InsectWingbeat (57%), JapaneseVowels (45%), and SpokenArabicDigits (57%), have missing values, of which the percentage of missing features is given in brackets. The missing values in these data sets are first filled with zeros. Given that the classifier performs better than random guessing on these data sets and the main aim of this work is to explain the classifier, filling missing values with zeros is acceptable. Second, the features of every dimension are normalised using the Min–Max normalisation method and then rescaled to [-1, 1]. The rescaling operation aims to keep the range of the MTS consistent with that generated by the generative model, of which the activation function in the final layer is Tanh.

Treating MNIST as an MTS: The original 28×28 images are treated as an MTS with 28 time steps, each of which has 28 features. Here, the y-axis is treated as the time steps and the x-axis as the features, which is shown in Fig. 3.4. Similarly, the images are normalised using the Min–Max normalisation method and then rescaled to [-1, 1].

3.4.2 Classifier

The novelty of this framework is the model-agnostic framework that explains any classifier through a post-hoc approach. Designing a high-performance MTSC is not our goal. Therefore, a simple classifier based on RNNs is im-

plemented for our experiments, where Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997) are used to encode the entire MTS and the hidden state at the final step are fed into a fully-connected layer followed by a soft-max layer to produce the probability of each class. The class with the highest probability is considered the predicted class. In our experiments, the dimension of the hidden states of this classifier is set to 128.

The accuracy of the classifier on MNIST is 98%, while the accuracy of the classifier varies among the data sets in the UEA archive. As shown in Table 3.2, the classifier performs excellently in some problems, such as JapaneseVowels, SpokenArabicDigits, and PenDigits. However, on other problems, such as AtrialFibrillation, EthanolConcentration and FingerMovements, its performance is not better than random guesses, where we assume that the classifier's outputs are not dependent on the inputs. In this circumstance, explaining their outputs does not make sense. Therefore, we only take data sets on which the classifier's performance is significantly better than random guesses in our experiments, which are shown in bold in Table 3.2. On thing that has to be denoted is that the data set in the UEA are all balanced, which means that each labels has the same number of samples which facilicate to judge whether the model performs better than random gauessing.

Among them, 17 data sets shown in bold meet this requirement. The samples in the test data sets are used in our explanation experiments. To save experiment time, for the problems InsectWingbeat, PhonemeSpectra, and SpokenArabicDigits, we randomly sample 1000 inputs from the test data sets to do experiments, while for other data sets, all the samples in the test data set are used in our experiments.

Table 3.2: The accuracy of the classifier and random guessing on the problems in UEA archive. The RNNs-based classifier performs well on the 17 data sets shown in bold.

	Data set	Accuracy			Data set	Accuracy	
		RNNs	Guess			RNNs	Guess
0	**ArticularyWordRecognition**	**0.89**	0.04	15	**InsectWingbeat**	**0.29**	0.10
1	AtrialFibrillation	0.33	0.33	16	**JapaneseVowels**	**0.94**	0.11
2	**BasicMotions**	**0.73**	0.25	17	**Libras**	**0.71**	0.07
3	**CharacterTrajectories**	**0.67**	0.05	18	**LSST**	**0.62**	0.07
4	**Cricket**	**0.79**	0.08	20	**NATOPS**	**0.86**	0.17
7	**Epilepsy**	**0.54**	0.25	21	**PenDigits**	**0.99**	0.10
8	EthanolConcentration	0.27	0.25	23	**PhonemeSpectra**	**0.14**	0.03
9	**ERing**	**0.79**	0.17	24	**RacketSports**	**0.76**	0.25
10	FaceDetection	0.59	0.50	25	SelfRegulationSCP1	0.53	0.50
11	FingerMovements	0.52	0.50	26	SelfRegulationSCP2	0.69	0.50
12	HandMovementDirection	0.34	0.25	27	**SpokenArabicDigits**	**0.98**	0.10
13	Handwriting	0.03	0.04	28	**StandWalkJump**	**0.47**	0.33
14	Heartbeat	0.70	0.50	29	**UWaveGestureLibrary**	**0.54**	0.13

3.4.3 Generative model

As outlined in our proposed framework, a generative model is utilized to learn the data distribution. Therefore, prior to generating explanations, the model is trained using training samples from each data set, and the hyperparameters of the generative model are tuned carefully. Specifically, the training data sets are used to train the model, and the test data set is used to evaluate the model's performance under various hyperparameter settings. The final tuned hyperparameters for the generative model are as follows:

- Number of fully connected layers (before the Transformer encoder): 2
- Dimension of hidden features in the fully connected layers: 128
- Number of Transformer layers: 2
- Dimension of the Transformer layers: 64
- Number of fully connected layers (after the Transformer encoder): 1

- Dropout rate for the fully connected layers and the Transformer layers: 0.2

3.4.4 Parameter Settings for the search algorithms

This section gives the parameter settings adopted for our experiments. In our experiments, the BPSO and GA methods are adopted for the identification steps in our proposed framework. The BPSO method is implemented using the package provided by (Miranda, 2018) and the GA method is implemented using the package provided by Ryan Solgi[1]. In our experiments, the parameters for these two search methods were tuned using the test data set. For each sample in the test dataset, we conducted multiple runs of the search algorithms with varying parameter settings. Our objective was to find the smallest supporting features. We evaluated the performance of each parameter setting based on the outcomes, selecting the best-performing set of hyperparameters as the final configuration. We chose to use only the test data set to tune the hyperparameters of our search algorithms. This decision is driven by the nature of the search methods, which, unlike traditional machine learning models, do not face overfitting issues. As long as the search results are satisfactory, the parameters are considered suitable. In addition, the final performance of the proposed method is evaluated only on the test dataset; thus, the test data set is sufficient to determine the optimal parameters for these search algorithms. The satisfying settings are shown below. For the BPSO method, the adopted parameters are:

- Population size: 100 (the number of particles in the swarm)
- Inertia weight (w): 1 (the influence of a particle's previous velocity

[1]https://github.com/rmsolgi/geneticalgorithm

on its current velocity)

- Cognitive coefficient (c1): 5 (the influence of a particle's personal best position on its velocity update)
- Social coefficient (c2): 5 (the influence of the swarm's global best position on the velocity update of a particle)

For the GA method, the adopted parameters are:

- Population size: 100 (the number of candidate in the population)
- Mutation rate: 0.2 (the probability of altering one or more genes in an individual's chromosome)
- Crossover rate: 0.5 (the probability of exchanging genetic material between two parent individuals during reproduction)
- Parents portion: 0.5 (the proportion of individuals in the population that are selected as parents for performing crossover and mutation operations to generate offspring)

Both the BPSO and GA methods used in our framework are strategically designed to ensure diverse starting points and effective exploration of the solution space. The process is organised as follows.

- Random Initialisation: Initially, the solutions for both methods are generated at random. This approach allows algorithms to explore various areas of the solution landscape, which is crucial to identify more robust global solutions.
- Setting Initial Solutions to All-Ones: The initial solutions for subsequent optimisation substeps are set to an "all-ones" state. This

technique is used because initial searches typically yield sufficiently small support features. By configuring solutions to all-ones, the algorithms are compelled to eliminate unhelpful or non-supportive features, thereby facilitating the discovery of more effective outcomes.

- Convergence Criteria: Both the BPSO and GA methods terminate when there is no improvement in the fitness function for 20 consecutive iterations. This threshold of 20 is strategically set to balance computational complexity with available computational resources, ensuring optimal usage without unnecessary computation.

3.4.5 Benchmarking methods

The explanation methods to be compared with are *Gradient-based methods*, *LIME* and *LIME-G*. These methods are widely adopted to explain classifiers and are taken as benchmarking methods in many works (Ismail et al., 2020; Crabbé and Van Der Schaar, 2021).

- *Gradient-based Methods*: The first gradient-based method we consider is SG, where importance scores are obtained by directly calculating the gradient of the probability of the predicted class w.r.t. the input features (Simonyan et al., 2014). In addition, we also consider the IG method, which numerically integrates gradients along a path between current input and a user-defined baseline (Sundararajan et al., 2017). Following the setting of (Sundararajan et al., 2017), we set the baseline in our experiments to zero input.

- *LIME*: LIME (Ribeiro et al., 2016) provides importance scores by learning a transparent proxy model in the local region near the input. It first perturbs the input data and creates a series of artificial

data. We realise it by randomly selecting the features of the input MTS and replacing them with zeros. Then, the classifier produces outputs for these artificial data. After that, a linear regression model is fitted, of which the input features are binary vectors indicating the 'presence' (by 1) or 'absence' (by 0) of the corresponding original features and the outputs are the probability score of the predicted classes. Finally, the absolute values of the weights of the fitted linear model are regarded as the importance scores of the corresponding features.

- *LIME-G*: The first step of LIME is to create a series of artificial data by perturbing the original input. This is achieved through zero replacement, which might create unrealistic inputs. Similar to (Agarwal and Nguyen, 2020), we also implement LIME-G, where G indicates that the artificial data set is created with the help of the designed generative model. Specifically, to create one perturbed input, we randomly select a number of original features to be perturbed and then generate possible alternative values for them using the generative model. Then, the original features are replaced by these alternative values to create a perturbed input. The other steps to obtain a feature importance map are the same as in the LIME method.

3.4.6 Quantitative Evaluation Metric

A good explanation should compactly identify supporting features for the outputs, which means that the classifier could produce a different output by replacing a minimal number of features with their alternative values. Here, we use the quantitative metric adopted in (Fong and Vedaldi, 2017; Chang et al., 2019) to evaluate the performance of a feature importance explana-

tion. Specifically, for a given explanation, we calculate the minimal number of features that are necessary to be replaced to change the classifier's output. The final explanations of the proposed method are binary. Therefore, this metric can be calculated directly by counting how many features are identified. However, the benchmarking explanation methods assign each feature an importance score. To make these explanations comparable to our method, we successively select features in order of their importance scores and then replace them with plausible values generated by the generative model until the classifier produces a different result. Whether the classifier produces a different result is estimated using (2.5), where $\boldsymbol{x}_r$ represents the features to be replaced. Finally, the selected features result in feature importance explanations in binary form. In this paper, we call this metric the Smallest Deletion Features (SDF). A smaller SDF means better explanation performance.

3.5 Results and Discussion

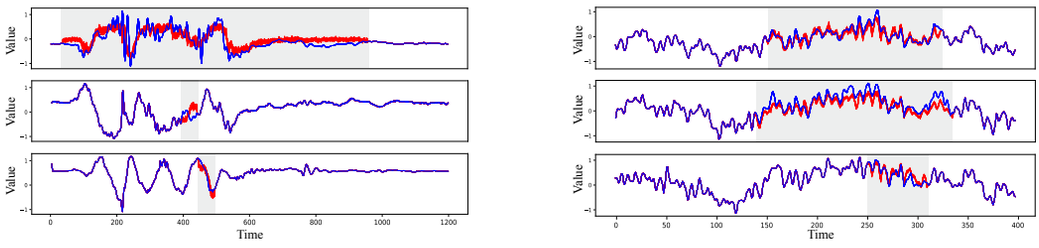
In this section, the performance of the proposed framework is systematically evaluated. First, the performance of the generative model in mitigating the OOD problem is evaluated quantitatively and qualitatively in Section 3.5.1, which is the basis for providing meaningful explanations. In Section 3.5.2, the performance of the adopted methods is quantitatively evaluated. However, with only feature importance explanations, users might not yet be able to understand the working mechanism of the classifier. Fortunately, our framework can provide counterfactual explanations, which will be described in Section 3.5.3. There are some stochastic processes in the proposed framework, but the explanations provided of a given classification should remain stable when we repeat the explaining process. Therefore,

in Section 3.5.4, a stability analysis of the proposed framework is carried out. Finally, in Section 3.5.5, ablation studies are carried out to analyse the contributions of the generative model and the proposed search strategy separately.

3.5.1 The performance of the generative model

The performance of the generative model is qualitatively evaluated, which is shown in Fig. 3.5. In Fig. 3.5a and Fig. 3.5b, the blue time series are the original time series, and the red subsequences are generated by the designed generative model. The results show that the generated values are close to the target. In Fig. 3.5c, we also compare the performance of the generative model in generating meaningful perturbation samples with that of traditional perturbation methods in the MNIST data set. To achieve this goal, some features of a sample are randomly selected and then perturbed by traditional methods and the designed generative model. For MNIST samples, the within-distribution inputs should look similar to digits written on the white background. It can be easily judged that the inputs created by the traditional methods do not meet this requirement. For example, noise is scattered over the background that should be white, or the original sharp features become blurry or even missing. In contrast, the generative model creates suitable values to replace the selected features, creating much more realistic samples, which suggests that it performs better in creating within-distribution perturbed inputs than traditional methods.

In addition to the qualitative evaluation above, we can also quantitatively evaluate the performance of the generative model in generating within-distribution perturbed samples. Inspired by (Hase et al., 2021), we cast this problem as a measurement of the robustness of the model to pertur-



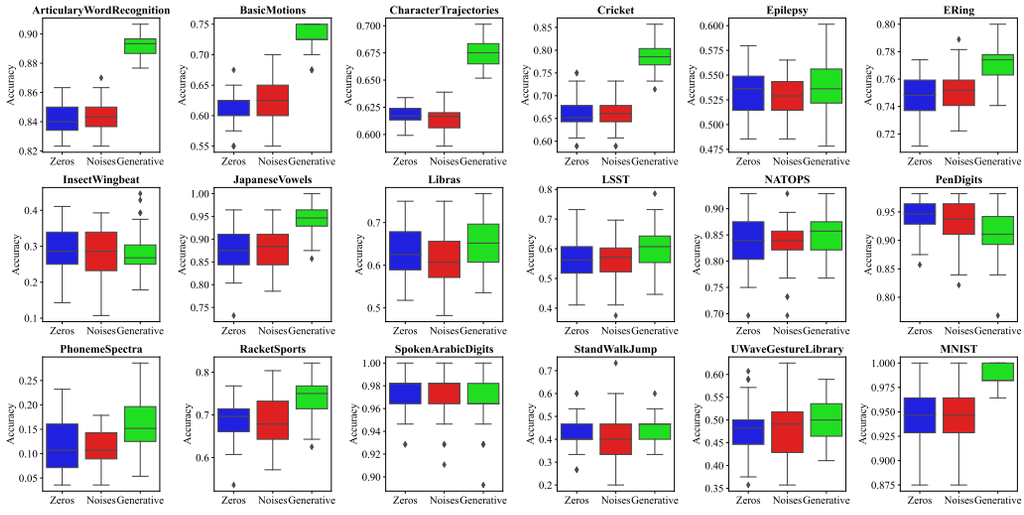
(a) Cricket

(b) HandMovementDirection

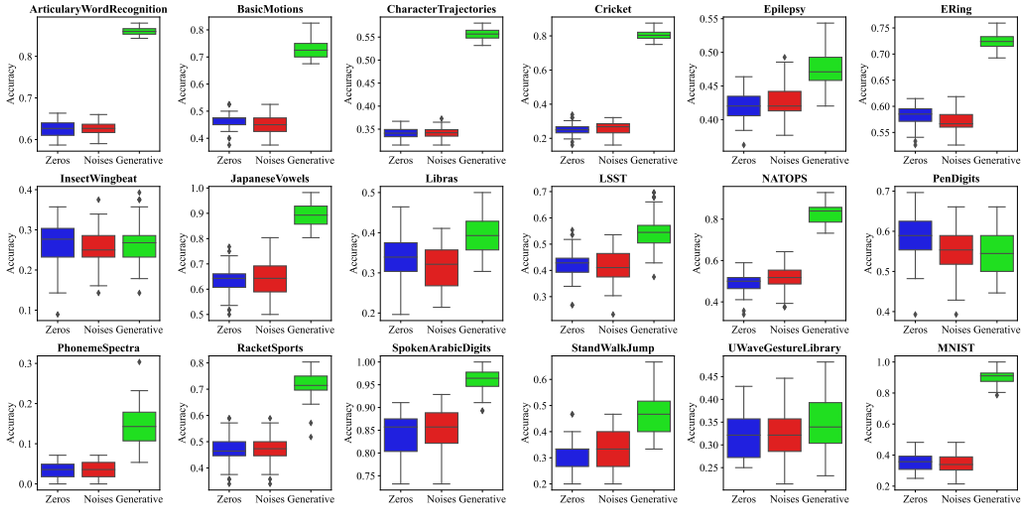
(c) MNIST

Figure 3.5: The performance of the generative model in creating plausible values for certain features. In (a) and (b), the blue time series are the original time series, and the red subsequences are generated by the generative model. In (c), the first and second columns are the original input and the input with some features randomly selected and masked. The third column to the last column are perturbed inputs created by the weighted average of neighbouring features (Crabbé and Van Der Schaar, 2021), by random noise, by the average values of corresponding dimensions and by the values created by the generative model, respectively.

bation, which is carried out as follows. For each input in the test data set, a certain percentage of features are randomly selected and then perturbed using zeros and random noise, as is common practice in the literature, and using the alternative values created by the generative model. The percentages of features to be perturbed here are {20%, 50%}. Then, the accuracy of the classifier on these perturbed inputs is measured, where the perturbed inputs are assumed to have the same labels as the original inputs. In this scenario, the performance of all classifiers is expected to deteriorate in line with increases in the percentage of perturbed features. However, we also expect that features replaced with random or arbitrary (e.g. zeros) replacement (which will tend to be OOD) will result in overall worse performance compared to features replaced with plausibale alternative values generated by the proposed generative model. The experiment is repeated 50 times for each percentage of features to perturb, and the predicted accuracy is



(a) 20% values are masked out



(b) 50% values are masked out

Figure 3.6: Model's prediction accuracy on the test data sets, when a certain number of features are perturbed using "zeros", "random noise", and alternative values generated by the generative model. Better performance in creating within-distribution inputs yields higher model's higher accuracy.

shown through boxplots in Fig. 3.6. Overall, the accuracy of the classifiers is higher on the perturbed inputs created by the generative model than on those perturbed using traditional approaches, which suggests that the classifiers are more robust to the perturbed inputs created by the designed generative model. This means that when some features are masked, the generative model could generate suitable alternative values on which the classifier could make correct predictions, such as the inputs created by the generative model in Fig. 3.5c. If these masked features are filled with zeros or noise, creating input that the classifier did not observe during the training stage, like those shown in Fig. 3.5c, the classifier's behaviour might be unpredictable and have a higher probability of making wrong predictions. However, the generative model does not always have a higher accuracy than traditional perturbation methods. For the InsectWingbeat data set, perturbation by zeros results in a higher accuracy than the generative model. The reason might be that there are many missing values in this data set (about 57%), and in our experiments they are filled with zeros. In this circumstance, perturbing with zeros might be a better choice. However, for other data sets, the generative model consistently performs better in generating within-distribution perturbed samples than traditional methods.

3.5.2 Quantitative Evaluation of Explanations

In this section, the SDF scores are calculated to quantitatively evaluate the performance of the explanation methods adopted. The results are shown in Table 3.3. The results show that our method achieves the smallest SDF scores, suggesting that fewer features are identified in the explanations provided. This helps to provide clear explanations.

Our framework is compatible with any binary search method, so we sep-

Table 3.3: The SDF results of the adopted saliency explaining methods (above) and the time required to obtain one explanation (below).

Data set	IG	SG	LIME	LIME-G	Proposed Framework	
					with BPSO	with GA
ArticularyWordRecognition	349.00(25.75) 0.02s(0.00)	401.50(20.17) 0.01s(0.00)	645.00(38.77) 5.30s(0.43)	591.65(40.38) 9.31s(0.87)	**83.50(7.23)** 27.59s(1.43)	89.00(7.50) 34.02s(1.85)
BasicMotions	86.00(6.00) 0.02s(0.00)	465.00(26.92) 0.01s(0.00)	125.00(8.32) 6.07s(0.56)	130.70(10.04) 10.45s(0.87)	73.50(4.86) 13.45s(0.90)	**60.00(3.22)** 14.08s(0.83)
CharacterTrajectories	112.00(9.81) 0.02s(0.00)	84.50(7.63) 0.01s(0.00)	108.50(7.82) 3.73s(0.20)	93.27(6.53) 6.51s(0.58)	32.50(2.55) 23.46s(1.58)	**28.00(2.75)** 24.21s(2.28)
Cricket	8.49(0.54) 0.08s(0.01)	22.23(2.18) 0.05s(0.01)	25.62(2.02) 29.26s(2.55)	29.30(2.29) 29.90s(2.50)	12.12(1.06) 142.43s(9.82)	**8.15(0.59)** 141.92s(11.15)
Epilepsy	8.00(0.48) 0.02s(0.00)	63.50(3.36) 0.01s(0.00)	7.50(0.41) 4.39s(0.29)	6.42(0.36) 6.89s(0.53)	**4.00(0.32)** 26.41s(1.89)	5.50(0.45) 29.80s(2.94)
ERing	26.50(2.64) 0.02s(0.00)	56.50(4.49) 0.01s(0.00)	16.00(1.45) 5.63s(0.36)	14.86(1.18) 10.92s(1.09)	13.00(0.83) 17.04s(1.16)	**12.00(0.67)** 7.29s(0.72)
HandMovementDirection	70.50(4.45) 0.03s(0.00)	86.50(5.49) 0.02s(0.00)	372.00(36.72) 8.56s(0.68)	364.44(29.79) 13.16s(1.25)	72.00(6.93) 102.22s(9.21)	**34.00(2.89)** 101.71s(6.97)
InsectWingbeat	19.50(1.93) 0.02s(0.00)	65.50(4.29) 0.01s(0.00)	23.50(2.11) 8.23s(0.67)	44.00(2.56) 14.85s(0.98)	15.50(1.18) 33.10s(9.79)	**14.02(0.77)** 31.08s(1.59)
JapaneseVowels	84.00(6.08) 0.02s(0.00)	164.50(14.09) 0.00s(0.00)	101.50(6.32) 7.89s(0.43)	88.67(8.09) 10.67s(0.96)	38.50(1.99) 43.57s(3.06)	**31.00(2.73)** 12.30s(0.93)
Libras	7.50(0.54) 0.03s(0.00)	18.00(1.02) 0.00s(0.00)	8.50(0.76) 4.93s(0.25)	7.74(0.65) 7.67s(0.63)	**6.00(0.46)** 5.06s(0.30)	7.00(0.36) 4.80s(0.25)
LSST	12.50(0.95) 0.02s(0.00)	52.00(3.60) 0.01s(0.00)	10.50(0.71) 5.18s(0.49)	8.46(0.68) 8.46s(0.48)	13.00(1.23) 7.22s(0.66)	**8.00(0.74)** 6.58s(0.50)
NATOPS	446.00(42.97) 0.03s(0.00)	656.00(48.58) 0.01s(0.00)	822.50(73.36) 10.66s(1.01)	818.67(50.49) 18.98s(1.29)	**111.50(8.38)** 10.59s(1.05)	128.00(11.78) 9.04s(0.71)
PenDigits	5.00(0.47) 0.01s(0.00)	11.00(1.08) 0.01s(0.00)	5.50(0.50) 3.17s(0.25)	5.43(0.27) 4.27s(0.38)	**3.00(0.26)** 2.12s(0.17)	**3.00(0.20)** 2.27s(0.20)
PhonemeSpectra	77.00(6.94) 0.02s(0.00)	153.50(9.71) 0.01s(0.00)	562.00(120.66) 8.09s(0.57)	632.27(82.72) 15.24s(1.48)	79.00(5.19) 50.25s(4.40)	**71.00(3.69)** 46.93s(3.90)
RacketSports	34.50(1.94) 0.03s(0.00)	116.50(7.01) 0.01s(0.00)	44.00(2.26) 4.32s(0.43)	41.61(2.69) 6.54s(0.63)	25.50(1.46) 4.93s(0.28)	**20.50(1.25)** 5.16s(0.31)
SelfRegulationSCP1	16.00(1.38) 0.03s(0.00)	**3.00(0.24)** 0.02s(0.00)	**3.00(0.24)** 9.22s(0.73)	3.04(0.29) 10.47s(0.58)	**3.00(0.22)** 35.51s(3.08)	**3.00(0.22)** 38.70s(3.21)
SpokenArabicDigits	219.00(14.77) 0.02s(0.00)	935.50(75.66) 0.01s(0.00)	638.00(39.57) 4.91s(0.44)	688.94(64.33) 6.68s(0.46)	115.00(8.01) 21.41s(1.42)	**93.50(5.86)** 21.91s(1.58)
StandWalkJump	10.50(0.66) 0.08s(0.01)	25.00(1.40) 0.06s(0.00)	29.00(1.61) 32.19s(2.13)	30.58(1.81) 32.44s(1.66)	12.00(0.68) 151.23s(14.26)	**8.00(0.44)** 164.26s(14.31)
UWaveGestureLibrary	27.00(1.61) 0.02s(0.00)	29.00(1.48) 0.01s(0.00)	36.00(3.26) 4.42s(0.34)	38.26(3.65) 8.38s(0.74)	**21.00(1.81)** 52.47s(3.50)	22.50(1.23) 56.89s(3.79)
MNIST	104.00(10.06) 0.02s(0.00)	167.00(15.72) 0.00s(0.00)	90.50(8.30) 4.01s(0.30)	95.51(6.70) 7.98s(0.76)	27.50(2.36) 9.43s(0.66)	**18.00(1.56)** 10.71s(0.96)

arately use the classic BPSO and GA methods in the identification steps of our framework. The results show that these two methods achieve very similar performance, suggesting that our framework is not sensitive to the choice of search methods. Therefore, users can adopt the search method they want in the identification steps.

From the aspect of time expense, gradient-based methods are very fast, since they only need back-propagation several times. But our framework needs much more forward propagation, and, besides, the generation of the perturbed inputs also takes time. Therefore, our method requires more time. However, for most of the problems here, our method can provide explanations within 20 seconds. For some applications where time constraints are not so strict, our method can be acceptable. However, if we take the quality of explanations into account, for example, in the MNIST data set, our method identifies about 30 important features. But the others identify nearly or more than 100 features. Therefore, it is worth sacrificing time for better explanations.

3.5.3 Counterfactual Explanations

By now, we have shown that our framework can provide explanations by pinpointing less important features than other methods, from which users can clearly identify the features that are relevant for the predicted class. However, only knowing which features are relevant is not enough. People might want to see how these relevant features would affect the classifier's output. If these features took other plausible values, what would the output of the classifier be? The answers to this question can provide more insight for ordinary people to understand the classifier (R. Fernández et al., 2022).

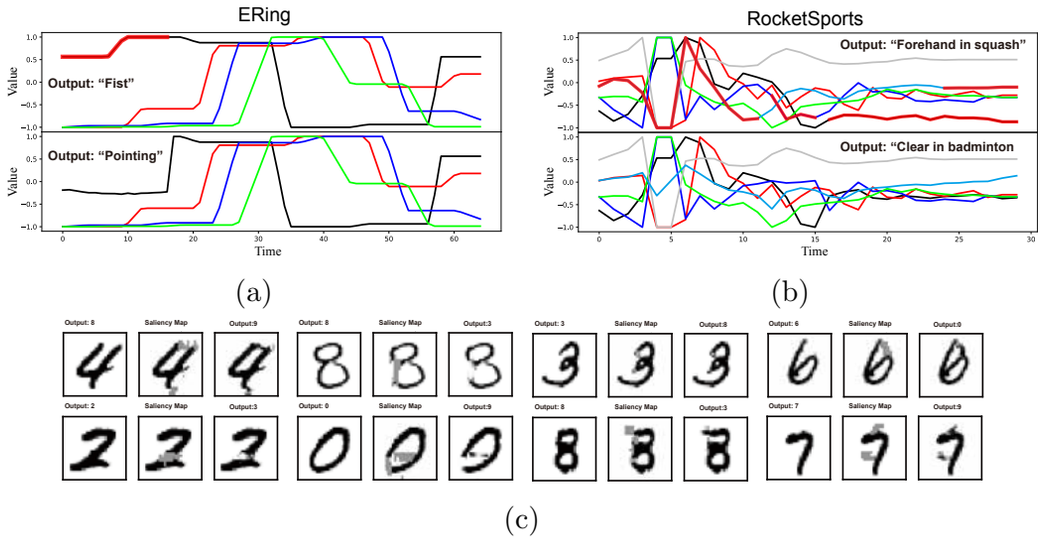


Figure 3.7: Counterfactual explanations for instances taken from ERing (a) and RocketSports (b) data set. The figures above are original time series, where the most important features are highlighted. The figures below are counterfactual inputs, where the important features take alternative values. (c): Counterfactual explanations for instances from the MNIST data set.

Fortunately, our framework can easily provide counterfactual inputs to answer this question. A counterfactual input is also a concrete MTS, which is very similar to the original input, but has a different class predicted by the classifier. The only difference between the counterfactual input and the original input is the features identified in the final explanations. Our framework can easily provide counterfactual inputs by replacing the identified features with alternative values generated by the designed generative model. Examples of counterfactual inputs are shown in Fig. 3.7. These counterfactual inputs can further explain the classifier by telling users how these features would affect the classifier's output. For example, if the identified features of input with the label '4', took large values (shown by dark pixels in the figure), the classifier would produce a label '9'.

3.5.4 Stability analysis

In this section, stability analyses of the explanation results are carried out. By stability, we mean that the explanations provided should be reproducible, which means that the explanations provided for a given classification should not vary too much if we repeat the explanation process multiple times. In our proposed framework, the inherent stochastic processes in both the heuristic search and the generation of perturbed inputs introduce apparent challenges in achieving stability. This aspect is demonstrated in Fig. 3.8, where we observe varying explanations for a single input from the CharacterTrajectories data set when the same explanation process is performed multiple times. Such variability, as demonstrated in the figure, underscores the influence of stochastic elements on the stability of our explanatory framework. However, we argue that this instability results mainly from a high number of degrees of freedom in the input space that can change the output of the classifier. For example, the input shown in Fig. 3.8a is predicted to be class '1'. However, the predicted class can be changed to one of the other 19 classes by perturbing different features. During the heuristic search, the stochastic processes make it uncertain which features are to be perturbed.

However, if the predicted label of the counterfactual explanation is constrained to a specific class, is the explanation stable? Specifically, which group of features of the input predicted as class '1', if replaced by other plausible values, can lead the classifier to produce a specific class '4'? Answers to this question can be easily obtained. We only need to modify the

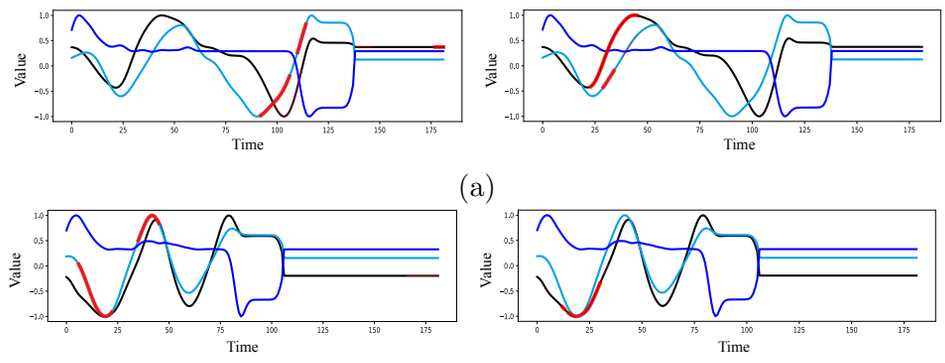


Figure 3.8: Different explanations are provided for the two inputs in the CharacterTrajectories data sets, where the most important features are highlighted by red lines.

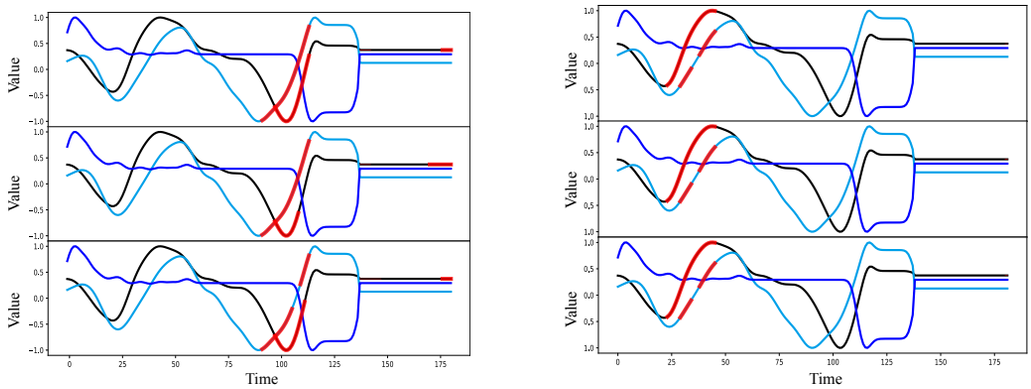


Figure 3.9: The saliency maps when the desired class is set to class "4" (a) and class "11" (c). The identified important segments are colored in red.

fitness function (3.4) to (3.9):

$$\text{Sup}(\boldsymbol{W}) = \begin{cases} 1 & \text{if } f(c|\boldsymbol{x}_{\setminus r}) \neq \text{desired class} \\ 0 & \text{if } f(c|\boldsymbol{x}_{\setminus r}) = \text{desired class} \end{cases} \tag{3.9}$$

$$\text{where } \boldsymbol{x}_{i,j} \in \boldsymbol{x}_{\setminus r}, \text{ if } \boldsymbol{W}_{i,j} = 1$$

Fig. 3.9 shows the counterfactual explanations when the desired class is specifically designed. It can be seen that the final explanations are much more stable than in Fig. 3.8.

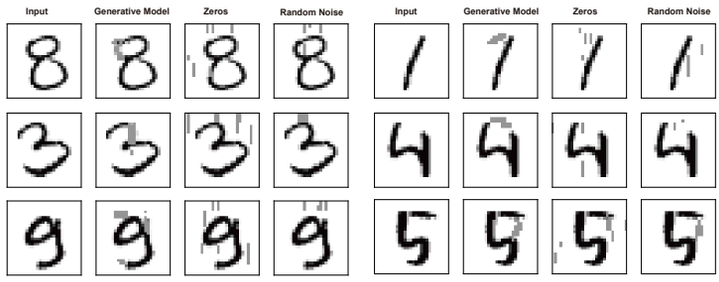


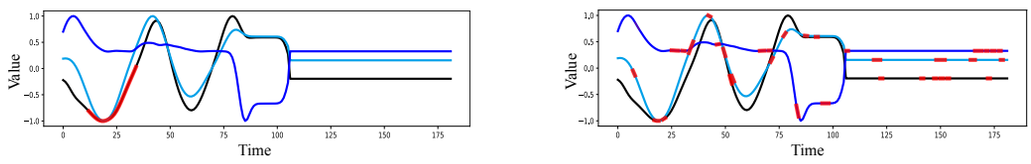
Figure 3.10: The saliency maps provided by our framework but using different perturbation strategies, including using the generative model, using zeros, and random noise.

3.5.5 Ablation studies

In this section, ablation studies are carried out to analyse the contributions of the two key components in the proposed framework, the designed generative model and the greedy-based segmentation and identification search strategy.

Contribution of the generative model. Fig. 3.10 shows the saliency maps provided for given classifications using our framework but with different strategies to generate perturbed inputs, including using the generative model and using traditional perturbation methods (zeros and random noise). The results show that the generative model really helps in generating meaningful explanations. The identified features are compact and located in the really meaningful regions rather than scattered among the meaningless background. These improvements are the result of the mitigation of the OOD problem.

Contribution of the greedy-based segmentation and identification strategy. In the proposed framework, a greedy-based segmentation and identification search strategy is designed for time series to mitigate the challenge of searching in a huge search space. The explanations provided using the proposed search strategy are compared with those provided with-



(a) (b)

Figure 3.11: The saliency maps for the given inputs (a) provided with our proposed search strategy and (b) provided by applying binary search method directly over the original features space.

out this search strategy. The latter explanations are provided by applying the same binary search algorithm (BPSO or GA) over the original feature space, where each feature is considered as one segment. As shown in Fig. 3.11, without the proposed search method, the identified features are scattered, while our method provides compact explanations, where the identified important features tend to be continuous in time steps. Because the proposed strategy considered features within continuous time steps as one 'super-feature', the final identified features also tend to be continuous in time. In addition, our search strategy is better in terms of sparsity. For example, on average, for the MNIST, only 38.33 features are identified as relevant with this strategy, while without our strategy, 69.09 features are identified. Moreover, this strategy can also notably speed up the explaining process: on average, for the MNIST, it takes about 20s to generate an explanation, but direct searching needs about 100s.

3.6 Summary

This chapter promoted the development of XAI in MTSC. Specifically, it proposed a model-agnostic framework that adopted a post-hoc approach to provide feature importance explanations in MTSC scenarios. Within this framework, two challenges were addressed. The first is the widely acknowledged OOD problem. The accurate distribution of the training

data set might not be accessible. Therefore, a generative model for MTS was designed to approximate the distribution of the data set to create perturbed inputs. The results of the experiments demonstrated that the classifiers were more resistant to perturbed inputs produced by the generative model than to those produced by traditional perturbation techniques. This suggests that the generative model was effective at producing within-distribution perturbed inputs. Another challenge comes from the huge search space, which increases exponentially with the number of features an MTS has. This challenge was addressed by a proposed greedy-based segmentation and identification strategy. As a result, the search space was significantly reduced, allowing classic search methods, including the BPSO and GA methods, to achieve satisfying results. The results of the ablation studies showed that the generative model and the proposed search strategy were necessary components to provide meaningful explanations. In addition to feature importance explanations, this framework has the potential to offer counterfactual explanations, which are more comprehensible to ordinary users and enhance their understanding of the classifier. Although the focus of this work was on MTSC problems, the proposed framework can be applied to any problem, such as those in image and language processing.

In this chapter, we concentrate on the precise identification of the most significant features, a crucial aspect of feature importance explanations. Moving forward, the next chapter will shift focus to another essential characteristic of feature importance explanation: its stability. Stability here refers to the consistency of explanations across multiple executions of explanation methods. A stable explanation method guarantees that, for a given task, the explanations it provides will always be consistent. Without stability, users may struggle to discern which insights are reliable, leading to confusion and mistrust. The current literature reveals that several

commonly used explanation methods, such as the widely recognised proxy method LIME, struggle to offer stable explanations (Zhou et al., 2021; Slack et al., 2021). Although some methods have been developed to enhance their stability, these methods, when applied in the MTSC context, do not provide stable explanations. Therefore, further research is imperative to uncover the reasons for this instability and provide more reliable explanations.

Chapter 4

Stable LIME Framework using a Generative Model and an Adaptive Weighting Method

4.1 Introduction

In the previous chapter, we introduced a novel framework designed to accurately highlight the most crucial features for feature importance explanations. This chapter seeks to address another critical issue associated with feature importance explanations: their stability. Stability refers to the consistency of explanations provided across multiple runs of an explanation method, an essential attribute for reliable feature importance explanations. Stable explanations allow model users to trust the results and take actions based on the insights provided.

In the existing literature, LIME stands out as a prominent explanation method, widely recognised and used in various applications. However, it

has been explicitly identified to have stability problems (Zhou et al., 2021; Visani et al., 2020; Slack et al., 2021). Efforts to improve its stability include increasing the number of neighbours (Zhou et al., 2021), optimising its hyperparameters (Visani et al., 2020), and improving the neighbour sampling process (Slack et al., 2021). Although these solutions have shown improved performance in tabular data, their effectiveness within the MTSC context falls short of expectations. The explanations remain notably unstable, significantly hindering their applicability in safety-critical areas. The root causes of this instability have not yet been fully understood, highlighting the need for further investigation.

Within the spectrum of methods aimed at enhancing the stability of LIME, a critical yet often overlooked aspect in these solutions is the method of neighbour generation. Common practices involve perturbing features by substituting them with noise or zeros, inadvertently leading to the creation of OOD samples. This OOD issue is a key factor contributing to the inconsistency of explanations. Addressing this issue is crucial for providing more stable and reliable explanations. To address this, we introduce SEGAL (**S**table **E**xplanations using a **G**enerative Model and an **A**daptive Weighting Method for **L**IME), a novel framework designed to enhance explanation stability. Specifically, in this framework, a generative model is used to create within-distribution neighbours, aiming to mitigate the OOD issue. Furthermore, an adaptive weighting method is proposed to facilitate the hyperparameter optimisation process within LIME. Through this innovative approach, SEGAL seeks to elucidate the underlying causes of LIME's unstable explanations in the MTSC context and offers more stable and reliable explanations.

The structure of this chapter is as follows: Section 4.2 provides a comprehensive description of the SEGAL framework, including the generation of

neighbours using generative models and the adaptive weighting method. Section 4.3 details the experimental design, while Section 4.4 presents the results and analysis. Finally, conclusions are drawn in Section 4.5.

4.2 Proposed method: SEGAL

In this section, a detailed description of the proposed method for **S**table **E**xplanations using a **G**enerative model and an **A**daptive weighting method for **L**IME (SEGAL), is provided. In order to address the challenge of generating within-distribution neighbours, we propose the use of a generative model, of which the details are described in Section 4.2.1. The second challenge is that the hyperparameters in the classic LIME framework significantly influence the stability of the final explanations and are hard to tune (Visani et al., 2020). To overcome this difficulty, a novel adaptive weighting approach with optimised parameters is designed and described in Section 4.2.2. An overview of the proposed method is shown in Fig. 4.1.

4.2.1 Creating neighbours using a generative model and local sampling

The core idea behind LIME is to interpret how a model behaves for a given input by analysing the model's responses to nearby samples, often referred to as neighbours. To be considered as neighbours, in this paper, we consider that there are two requirements that samples must meet. First, they should come from the same distribution as the training data set. Otherwise, they might not be realistic and cannot represent what the model has learnt from the data set, and therefore should not be used to provide expla-

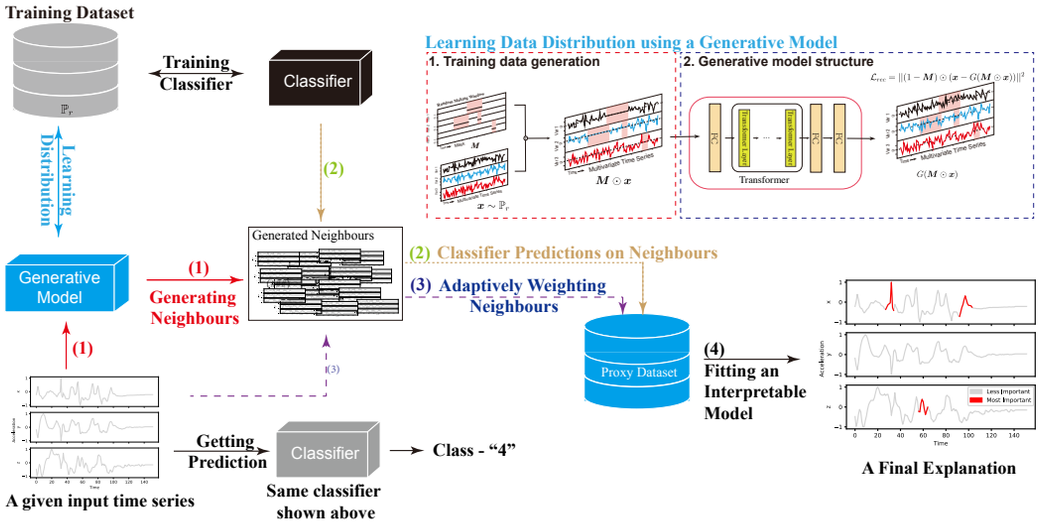


Figure 4.1: The workflow of the proposed method. Prior to generating explanations, the classifier and a generative model are trained on the data set. The details of training the generative model are shown on the top right. The steps to explain the classifier prediction for the given input time series (also known as the target sample to be explained) are as follows: (1) Generation of neighbours for the given input using the trained generative model; (2) Making of predictions on the generated neighbours using the trained classifier; (3) Weighting of the created neighbours using the proposed adaptive weighting method; (4) Fitting of an interpretable model on the proxy data set to provide a final explanation.

nations (Hase et al., 2021; Meng et al., 2023). Second, they must be located close to the target to provide local explanations. However, the traditional neighbour generation process used in LIME does not meet the above requirements. To mitigate this shortcoming, a generative method combined with a local sampling approach is proposed. The generative model maintains the generation of within-distribution samples, and the local sampling allows the within-distribution samples to be close to the target. The details of these two parts are explained as follows.

Creating within-distribution samples. As illustrated above, it is crucial that the neighbour samples used in LIME are drawn from the same distribution as the data set upon which the model is trained. If the distribution of the data set were readily available, the acquisition of samples within the distribution would be straightforward. However, in practical

applications, the reality is that gaining access to the precise distribution of the real-world process underlying the data set is often impossible. As presented in Chapter 3, we propose to use a generative model to approximate the underlying distribution of the training data set.

In particular, the aim of the distribution estimation is to generate plausible alternative values for certain features, eliminating their contributions from the model's output via a marginalisation operation (Kim et al., 2020; Chang et al., 2019):

$$\begin{aligned} f(c|\boldsymbol{x}_{\setminus r}) &= \int f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r}) d\boldsymbol{x}_r^* \\ &= \mathbb{E}_{\boldsymbol{x}_r^* \sim p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r})}[f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r})] \end{aligned} \tag{4.1}$$

where $\boldsymbol{x}_r$ denotes a subset of the input features, and $\boldsymbol{x}_{\setminus r}$ denotes its complementary part, $\boldsymbol{x} = \boldsymbol{x}_r \cup \boldsymbol{x}_{\setminus r}$; $f(c|\boldsymbol{x}_{\setminus r})$ denotes the classifier's output when the contributions of $\boldsymbol{x}_r$ are removed. The difference between $f(c|\boldsymbol{x}_{\setminus r})$ and $f(c|\boldsymbol{x})$ measures the contributions of $\boldsymbol{x}_r$. $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$ in Eq. (4.1) describes the distribution of plausible values of $\boldsymbol{x}_r$ conditioned by $\boldsymbol{x}_{\setminus r}$. Therefore, the objective of the generative model here is to estimate this conditional distribution, $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$.

The similar idea of using a generative model to approximate the distribution of the training data set and estimate the values of $f(c|\boldsymbol{x}_{\setminus r})$ has been adopted in (Chang et al., 2019; Meng et al., 2023). This idea follows the following steps:

Training data generation. To train a generative model to estimate the distribution, $p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r}^*)$, the first step is to obtain its inputs, $\boldsymbol{x}_{\setminus r}^*$, and outputs, $\boldsymbol{x}_r^*$. As shown in Fig. 4.1 we sample time series data from the data set, and for each data, a binary mask is created to mask out certain features through element-wise multiplication, $\boldsymbol{x}^* \odot \boldsymbol{M}$, creating an input for the

generative model. The masked components, $\boldsymbol{x}^* \odot (\mathbf{1} - \boldsymbol{M})$, are the target that the generative model tries to generate.

Architecture of the generative model. In this work, the generative model is designed based on the Transformer (Vaswani et al., 2017). The details of the model structure are shown in Fig. 4.1, which is very similar to that adopted in Chapter 3. The input time series data are processed by a fully connected layer, which transforms it into a high-dimensional feature space to better capture the temporal denpendency. This is followed by a series of Transformer layers that comprise the Transformer encoder. Subsequently, the output from the Transformer encoder is directed to another fully connected layer, which includes a hyperbolic tangent (tanh) activation function. This function is specifically used to ensure that the predicted values are within the range of [-1, 1].

Loss function. The generative model aims to produce the masked component. Therefore, the loss functions adopted here is a reconstruction loss:

$$\mathcal{L}_{rec} = ||(1 - \boldsymbol{M}) \odot (\boldsymbol{x} - G(\boldsymbol{M} \odot \boldsymbol{x}))||^2 \tag{4.2}$$

where G denotes the whole generative model. This loss measures the distance between the masked components of the original inputs and those generated by the generative model.

Estimating $p(c|\boldsymbol{x}_{\setminus r})$ using Monte Carlo method. In our approach, we use the Monte Carlo method to estimate $p(c|\boldsymbol{x}_{\setminus r})$ as given in (4.1). To achieve this, it is necessary to obtain samples from the distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$. The generative model we propose is capable of predicting the values of $\boldsymbol{x}_r$ when conditioned on $\boldsymbol{x}_{\setminus r}$. As demonstrated by (Gal and Ghahramani, 2016), maintaining the dropout layers active in the generative model during infer-

ence and repeatedly executing the forward process is identical to drawing samples from the distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$. Thus, using a reparameterisation trick, we can reformulate (4.1) as follows:

$$\begin{aligned} f(c|\boldsymbol{x}_{\setminus r}) &= \int f(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\setminus r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\setminus r}) d\boldsymbol{x}_r^* \\ &= \int f(c|G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r}), \boldsymbol{x}_{\setminus r}) d\boldsymbol{\epsilon} \\ &= \mathbb{E}_{\hat{\boldsymbol{\epsilon}} \sim p(\boldsymbol{\epsilon})} f(c|G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r}), \boldsymbol{x}_{\setminus r}) \end{aligned} \tag{4.3}$$

where $\hat{\boldsymbol{\epsilon}}$ is a product of Bernoulli distribution with probabilities of $1-p$; p represents the dropout rate in the training stage; the $\boldsymbol{\theta}$ represents the weights of the final fitted generative model. Keeping the dropout layers active is equivalent to sampling $\hat{\boldsymbol{\epsilon}}$ from the distribution $p(\boldsymbol{\epsilon})$. To calculate $f(c|\boldsymbol{x}_{\setminus r})$, it is necessary to run multiple forwards of the generative model, which is very time consuming, especially for large models. Therefore, in this work, the expectation of the plausible alternative values is used to estimate the values of $f(c|\boldsymbol{x}_{\setminus r})$, which is also adopted in (Chang et al., 2019) and can be represented as:

$$\begin{aligned} f(c|\boldsymbol{x}_{\setminus r}) &= \mathbb{E}_{\hat{\boldsymbol{\epsilon}} \sim p(\boldsymbol{\epsilon})} f(c|G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r}), \boldsymbol{x}_{\setminus r}) \\ &\approx f(c|\mathbb{E}_{\hat{\boldsymbol{\epsilon}} \sim p(\boldsymbol{\epsilon})}[G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}(\boldsymbol{x}_{\setminus r})], \boldsymbol{x}_{\setminus r}) \\ &= f(c|\overline{\boldsymbol{x}_r'}, \boldsymbol{x}_{\setminus r}) \end{aligned} \tag{4.4}$$

where $\overline{\boldsymbol{x}_r'}$ denotes the expectations of the generated alternative values for the features to be removed. Consequently, $\overline{\boldsymbol{x}_r'} \cup \boldsymbol{x}_{\setminus r}$ represents one within-distribution sample. In this particular sample, the feature set denoted by r is the expectation of the alternative values generated, while the remaining features retain the original values. The value of $\overline{\boldsymbol{x}_r'}$ can be obtained by performing a forward propagation of the generative model with the dropout layers deactivated, which improves the efficiency of the proposed method.

Creating neighbours with a local sampling approach. The objective of the generative model above is to generate within-distribution samples. However, it is also necessary for the generated within-distribution samples to be close to the target to be considered neighbours. In Section 2.4.3, it has been mentioned that creating a neighbour involves two steps: sampling neighbours in the interpretable feature space, where the features are '0' or '1', and recovering the sampled neighbours in the original feature space, where the features are real values. Traditionally, when sampling an interpretable representation, each feature is independently sampled from a Bernoulli distribution with a probability of 0.5. However, as will be illustrated in Section 4.4.1, this sampling approach cannot ensure that generate neighbours close to the target. To address this shortcoming, we propose a local sampling approach to sample the interpretable representation, which takes the following two steps:

- First, a percentage value, p, is randomly sampled from a uniform distribution, $p \sim \mathcal{U}(0.5, 1.0)$.
- Then, each interpretable feature is independently sampled from a Bernoulli distribution with a probability of p[1].

After an interpretable representation of the neighbour is obtained, the generative model is used to recover it into the original representation. Specifically, the features indicated by '0' in the interpretable representation are masked out from the original time series. The generative model is then used to generate values that effectively impute these masked features, recovering this neighbour in the original feature space.

[1]Traditionally, the p is usually fixed to 0.5.

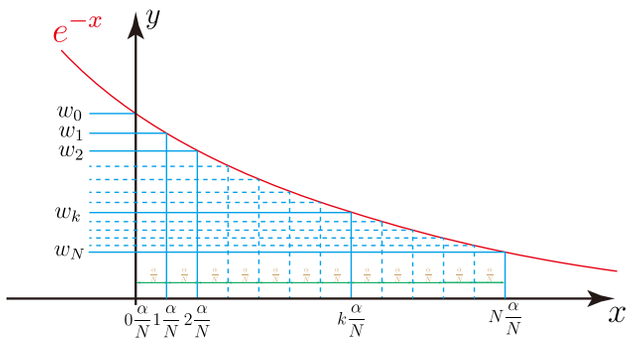


Figure 4.2: The proposed weighting method assigns the highest weights to the closest neighbor, and the weights decrease exponentially as the neighbors are farther away.

4.2.2 Adaptive weighting method

In (Visani et al., 2020), the authors showed that σ in the exponential kernel, which is $\pi_x(\boldsymbol{z}') = exp(-D(\boldsymbol{x}', \boldsymbol{x}^t)/\sigma^2)$, plays a significant role in the stability of the explanations. Therefore, they propose to search for a suitable σ using an optimisation method. To optimise σ, the first step is to define an appropriate search interval to find the optimal σ. Given the definition of the exponential kernel, an appropriate value of σ can be expected to depend on the absolute distances between the target and its neighbours. These absolute distances might vary substantially across different problems or even amongst various instances of the same problem. Consequently, the ideal σ can fluctuate considerably from instance to instance, complicating the task of defining a suitable search interval for the optimisation method.

Since LIME's weighting strategy aims to assign higher weights to closer neighbours, it seems logical to develop a method that adaptively allocates larger weights to closer neighbours, irrespective of the absolute distance measurement. Thus, we propose an adaptive weighting method, which always assigns the highest weight, for example '1', to the closest neighbours and progressively reduces the weights for neighbours further from the target. This novel weighting strategy is shown in Fig. 4.2 and can be

articulated as follows:

$$w_k = \exp(-\alpha \frac{k}{N}) \tag{4.5}$$

where N denotes the total number of neighbours created during the explaining process, and k represents the rank of the distance from a specific neighbour to the target. In this thesis, the distance is measured using dynamic time warping. The nearest neighbour has a rank of 0. The weight w_k is assigned to the k-th closest neighbour. The parameter α, known as the adaptive weighting scalar, determines the range of neighbours considered in the explanation process. For example, when $\alpha = 0$, all neighbours have identical weights, which means that all neighbours are taken into account. As α increases to a large value, only the most closed neighbours among all created neighbours are assigned non-zero weights. This indicates that only a small range of neighbours is considered. Similarly to σ in (2.6), in the proposed method, α also needs to be optimised. However, the appropriate value of α depends on the relative distances between the neighbours and the target, ignoring the absolute distances. For a given problem, it can be optimised within the same search interval.

As described in (4.5), the importance assigned to the samples decreases exponentially with their distance rank from the target. This setting ensures that only the nearest neighbours are considered significant. This nonlinear decrease design is based on an important consideration: samples that are very close to the target are less likely to be OOD. As distance increases, the likelihood of encountering OOD samples also increases. Thus, this nonlinear decay effectively mitigates the influence of OOD samples. Moreover, such exponential decay ensures that the importance assigned to any sample can never be zero or negative.

Table 4.1: Summary of data set characteristics and performance of the adopted classifiers and generative model. The column, $D \times T$, represents the total number of features when fitting interpretable models.

Problem	D × T	Accuracy		Generative Loss
		Transformer	LSTM	
ArticularyWordRecognition	1296	0.9933	0.8400	0.0570
BasicMotions	600	0.9250	0.6250	0.1095
CharacterTrajectories	546	0.9889	0.6602	0.0890
Epilepsy	618	0.6087	0.5290	0.2021
ERing	260	0.9296	0.6704	0.2765
Handwriting	456	0.2506	0.0659	0.0741
JapaneseVowels	348	0.9595	0.8919	0.0573
Libras	90	0.8056	0.6556	0.3354
LSST	216	0.6111	0.6135	0.1445
NATOPS	1224	0.8722	0.8444	0.0510
PenDigits	16	0.9866	0.9874	0.3325
RacketSports	180	0.7763	0.7368	0.1134
SpokenArabicDigits	1209	0.9909	0.9850	0.0447
UWaveGestureLibrary	945	0.8750	0.4250	0.1641
MNIST	784	0.9998	0.9980	0.0321

4.3 Experimental Design

In this section, we describe in detail the experimental design of this chapter. First, the adopted data sets and classifiers are described in Section 4.3.1. Then, in Section 4.3.3, benchmarking methods are introduced and their implementation details are given. Finally, a quantitative evaluation metric for the stability of explanations is provided in Section 4.3.4.

4.3.1 Data sets and classifiers

To assess the performance of our proposed framework, we selected 15 data sets from the UEA time series classification archive (Bagnall et al., 2018). We limited our experiments to datasets with fewer than 2000 features due

to our computational constraints[2]. Details of the selected data sets are provided in Table 4.1. Some data sets in the archive contain a large number of time steps, which significantly increases memory requirements during the distribution approximation process with our generative model[3]. Additionaly, certain data sets present an extensive number of features, resulting in not only very large memory requirements during the fitting of the interpretable model, but also making the explanation process very time-consuming[4]. The handwriting image data set, MNIST (Lecun et al., 1998), is also adopted. The main goal of using the MNIST data set is to facilitate a qualitative evaluation of explanations. For the adopted data sets, we use the original train/test split. The training samples are used to train the classifiers and the proposed generative model, while the test samples are used to evaluate the performance of explanation methods.

In this work, two currently popular black-box classifiers are implemented: LSTM-based (Hochreiter and Schmidhuber, 1997) and Transformer-based (Vaswani et al., 2017).

- ***LSTM-based***: This is one of the widely adopted recurrent neural networks to address time-series data. LSTM cells are used to encode the entire time series, and the hidden states at the final step are fed into a fully connected layer, followed by a softmax layer, to make

[2]All computational experiments were conducted on a machine equipped with an Intel(R) Xeon(R) Bronze 3104 CPU, 64 GB of RAM, and an NVIDIA Titan V GPU.

[3]The core of a Transformer model, the backbone of our generative model, is the attention mechanism, which calculates the importance of each point in the sequence to every other point. In terms of memory, this means that the model must store a matrix of size T-by-T, where T is the number of time steps. Therefore, the memory requirement for these models scales quadratically with the sequence length

[4]In the process of fitting the linear model, we use the analytical solution to derive the coefficients for the respective features, expressed as $\beta = (X^T X)^{-1} X^T y$. This process requires the creation of an n-by-n matrix, where n represents the number of features. With an increasing number of features, the size of this matrix can increase significantly, thereby requiring extensive memory resources. Furthermore, the inversion of the matrix, $(X^T X)^{-1}$, is computationally demanding, involving a complexity of approximately $O(n^3)$.

classification predictions. The main hyperparameters of this model are as follows: the hidden state dimension is 128, and the model consists of a two-layer LSTM architecture

- ***Transformer-based***: This model adopts self-attention mechanisms to capture temporal dependencies and has shown remarkable performance in processing time series data. In this work, the Transformer encoder is used to encode the entire time series, followed by a softmax layer for making final predictions. In our experiments, the model has a dimension of 128, a feed-forward layer dimension of 256, 4 encoder layers, and 8 attention heads.

The LSTM-based model was chosen for its ability to sequentially process time series data and efficiently handle long-term dependencies. In contrast, the Transformer model, which processes entire sequences concurrently, excels at capturing both local and global dependencies. Both models have demonstrated strong performance in multivariate time series classification tasks. Therefore, it is interesting to investigate the performance of the proposed method in these two kinds of models. The statistical information of the selected data sets and the classification performance of these two classifiers are shown in Table 4.1.

4.3.2 Generative model

As outlined in our proposed framework, a generative model is utilized to learn the data distribution. Therefore, prior to generating explanations, the model is trained using training samples from each data set. The performance of the generative model is then evaluated using test samples. The generative losses, as defined in (4.2), are shown in the last column of Ta-

ble 4.1. The hyperparameters of the generative model are tuned carefully. Specifically, the training data sets are used to train the model, and the test data set is used to evaluate the model's performance under various hyperparameter settings. The final tuned hyperparameters for the generative model are as follows:

- Number of fully connected layers (before the Transformer encoder): 2
- Dimension of hidden features in the fully connected layers: 128
- Number of Transformer layers: 2
- Dimension of the Transformer layers: 64
- Number of fully connected layers (after the Transformer encoder): 1
- Dropout rate for the fully connected layers and the Transformer layers: 0.2

4.3.3 Benchmarking Explanation Methods

In order to demonstrate the effectiveness of the proposed method, the key works in the literature described in Section 2.4.4 are implemented and compared with the proposed method. A summary and implementation details are shown below.

- ***LIME*** (Ribeiro et al., 2016): This serves as the baseline method. The neighbours are created and weighted according to the default settings, shown in Table 2.1.
- ***Slime*** (Zhou et al., 2021): This method begins with a small number of neighbours and gradually adds more neighbours to the proxy data

Table 4.2: Number of neighbours for the adopted methods.

	Number of neighbours
LIME	10000
Slime	Initial: 500; max: 20000
BayesLIME	Initial: 500; max: 10000 increase 500 for every iteration
OptiLIME	10000
SEGAL (ours)	10000

set until the given hypothesis test is passed or the maximum number of neighbours is reached. Because in this work the performance of these methods is compared based on the k most important features, in Slime, the hypothesis test adopted is to determine whether the k most important features remain the same. Similar to the original work, the confidence level for the hypothesis test is set to 0.95.

- ***BayesLime*** (Slack et al., 2021): This method proposes focused sampling to efficiently sample interpretable representations of neighbours (see Section 2.4.3).

- ***OptiLIME*** (Visani et al., 2020): This method requires optimising σ in (2.6). In our work, we employ Bayesian optimisation to find the best σ. The optimisation objective of the search method is to maximise the stability evaluation metric for explanations, which will be described in Section 4.3.4.

For the above and the proposed methods, the number of neighbours adopted in the explaining process is an essential hyperparameter. The setting of this parameter is shown in Table 4.2. The approach proposed by Slime involves adding more neighbours should the test fail. Given that 10,000 neighbours, which are set to other methods, might be insufficient for the test to pass, the maximum number of neighbours employed by Slime is doubled in comparison to other methods.

In all instances of the LIME framework adopted for this study, a linear regression model serves as the interpretable model. The inputs to this model are features in the interpretable feature space, denoted by 0 or 1. The output of the linear model is the probability of the target label. Linear regression is usually chosen in the LIME framework for its simplicity and effectiveness in providing clear, understandable relationships between features and the target outcome.

In OptiLIME and our method, it is necessary to optimise the hyperparameters, that is, σ in Eq. (2.6) and α in Eq. (4.5). Similar to the work (Visani et al., 2020), Bayesian optimisation is used to achieve this goal. We carried out preliminary experiments and defined the following search intervals. It is $[1e^{-4}, 0]$ for σ and $[0, 80]$ for α. Given the small suitable interval for σ, the search interval is logarithmically distributed. We take the 10 starting values evenly distributed over the corresponding intervals. Subsequently, the stability of the explanations for these starting values was evaluated. Then, an iterative process was initiated to find the satisfying hyperparameters. We established a maximum iteration cap at 10, a number that we found to be sufficient to obtain very stable explanations for the majority of the adopted problems. For example, the final stability metric exceeds 0.9. During this iteration process, a stability evaluation metric was measured (the details of which will be explained in Section 4.3.4). If this metric exceeds 0.9, the iteration process will be terminated early to save time. The number of iterations carried out is noted and used to evaluate and analyse the performance of search efficiency in Section 4.4.2.

In addition, we have implemented two variants of the proposed method to investigate the contributions of each component in detail, the details of which are listed in Table 4.3. SEGAL represents the complete proposed method which incorporates the use of the generative model and the

Table 4.3: Method variants for detailed analysis

Method	Neighbours generation		Weighting method
	Sample interpretable neighbours	Recover into real neighbours	
SEGAL	$p \sim \mathcal{U}(0.5, 1.0)$ iid $\sim$ Bern (p)	Generative	(4.5) with optimised α
SEGKL	$p \sim \mathcal{U}(0.5, 1.0)$ iid $\sim$ Bern (p)	Generative	(2.6) with optimised σ
SETKL	$p \sim \mathcal{U}(0.5, 1.0)$ iid $\sim$ Bern (p)	Traditional*	(2.6) with optimised σ

*Traditional: replace features with the average of the corresponding variables.

adaptive weighting method, while SEGKL exclusively adopts the proposed generative model. The objective of SEGKL is to evaluate the contributions of the proposed weighting method. In addition, SETKL is implemented, which adopts the proposed local sampling method in the sampling of interpretable neighbours, but utilises the traditional neighbour generation and weighting approach. This method will be compared with SEGKL to investigate the contribution of the generative model.

4.3.4 Stability Evaluation Metric

To evaluate the stability of explanations, it is necessary to consider which types of explanation are stable from the perspective of the model users. Users may focus only on the most important features, which means that as long as the most important features identified remain consistent among a set of explanations, these explanations are stable. For example, consider two explanations, of which the top-k most important features are set A and set B, respectively. If the similarity between these two sets is high, it indicates that these two explanations are stable for users. Therefore, we use the same evaluation metric as adopted in Slime (Zhou et al., 2021).

Specifically, the Jaccard index is used to measure the similarity between the identified most important features. It is defined as the size of the intersection divided by the size of the union of those two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.6}$$

In this work, the same explanation process is carried out 20 times, as conducted in (Zhou et al., 2021), for each instance to be explained. For each explanation, its important feature set is composed of the top-k most important features, selected by ranking the features according to the absolute values of their importance scores. Subsequently, the Jaccard index is used to assess the similarity of these important feature sets. A higher Jaccard index means more stable explanations. Here, the top 1% of the most important features of the explanations are used to assess the stability.

In this study, we employ the Wilcoxon Signed-Ranks test (Conover, 1999) to analyse the performance differences between the benchmarking methods and the proposed one in detail. We set the significance level for the tests at α (0.05) (Holm, 1979), a standard choice in many scientific studies.

4.4 Results and analysis

In this section, details of the experimental results are provided. First, Section 4.4.1 shows that the proposed method generates within-distribution neighbours for LIME. Then Section 4.4.2 demonstrates the effectiveness of the proposed adaptive weighting method. Section 4.4.3 compares the proposed method with benchmarking methods. Finally, in Section 4.4.4, the contributions of the proposed generative method on the stability of explanations are analysed.

Table 4.4: Methods for creating samples for LIME

Method	Neighbours generation	
	Sample interpretable neighbours	Recover into real neighbours
Traditional method	iid ∼Bern (0.5)	Traditional
Generative method	iid ∼Bern (0.5)	Generative
Generative method with local sampling	$p \sim \mathcal{U}(0.5, 1.0)$ iid $\sim$ Bern (p)	Generative

*Traditional: replace features with the average of the corresponding variables.

4.4.1 Creating within-distribution neighbours

As mentioned in Section 4.2.1, neighbours should be located in close proximity to the target to be used in the explanation process. However, the traditional method described in Section 2.4.3 does not meet this requirement. In order to demonstrate this, we generate 10,000 neighbours for every test sample of the adopted data sets using the traditional method, as well as two generative methods, one employing solely the generative model and the other integrating the local sampling strategy. Details of these three methods are shown in Table 4.4. The average distances, measured by cosine similarity, between the created neighbours and the corresponding target samples are shown in Table 4.5. Numbers in bold indicate that the corresponding method generates closer neighbours than the other methods.

For a more detailed investigation, in Fig. 4.3, we visualise an example of the distance distributions between a target input selected from the JapaneseVowels data set and the correspondingly generated 10,000 neighbours, using all three methods. The results show that traditionally created neighbours are significantly further from the target, highlighting the challenge in using them to derive explanations that are relevant to the input sample in question. In contrast, generative approaches create neighbours that

Table 4.5: Average distance between created neighbours and target samples across the adopted data sets.

Data set	Traditional method	Generative method	Generative method with local sampling
ArticularyWordRecognition	0.2554	0.0740	**0.0251**
BasicMotions	0.2186	0.1770	**0.0777**
CharacterTrajectories	0.2426	0.1275	**0.0462**
Epilepsy	0.2514	0.2417	**0.1113**
Ering	0.2422	0.1619	**0.0682**
Handwriting	0.2204	0.2062	**0.0931**
JapaneseVowels	0.2414	0.0513	**0.0176**
Libras	0.2699	0.2358	**0.0995**
LSST	0.1842	0.1575	**0.0710**
NATOPS	0.2602	0.0483	**0.0197**
PenDigits	0.2953	0.2577	**0.0986**
RacketSports	0.1841	0.1449	**0.0629**
SpokenArabicDigits	0.2105	0.0766	**0.0297**
UWaveGestureLibrary	0.2676	0.1596	**0.0602**
MNIST	0.0942	0.0232	**0.0082**

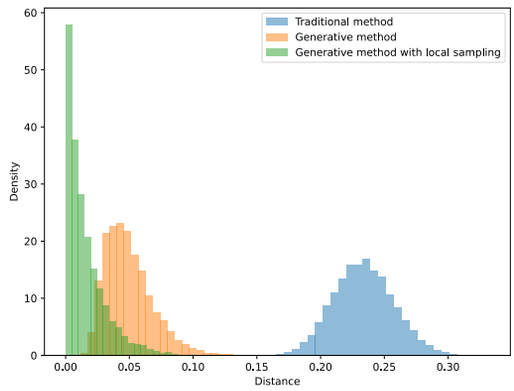
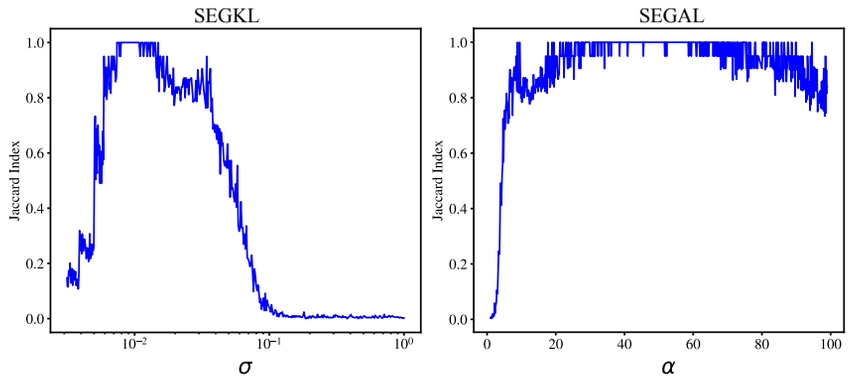


Figure 4.3: The distance distributions, measured by cosine similarity, between the target input and neighbours created by three different methods. The target sample is taken from the JapaneseVowels data set.

are significantly closer to the target than those created by the traditional method. However, using the generative method alone (where the features of the interpretable neighbours are independently sampled from a Bernoulli distribution with a probability of 0.5), the created neighbours tend to 'surround' the target, which is shown by a low density in the immediate vicinity of the target. However, for LIME, the ideal situation is to have neighbours that 'overlap' the target, which enables these neighbours to represent the behaviours of the models on the local region of the target. As depicted in



(a) (b)

Figure 4.4: The influence of σ in SEGKL (a) and α in SEGAL (b) on the stability of final explanations. If these two hyperparameters are not chosen suitably, the final explanations will not be stable (This sample comes from the Handwritting data set and the classifier is Transformer-based.)

Fig. 4.3, the proposed generative method combined with the local sampling method mitigates this shortcoming, resulting in a large number of neighbours that cover the target. One thing that should be denoted is that the results depicted in Fig. 4.3 can be somewhat misleading. It appears that many samples have a zero distance to the target. However, in reality, the distance between the created samples and the target can never be zero; it is merely very close to the target. This approach achieves our goal of creating very close neighbours to the target, facilitating the generation of local explanations.

4.4.2 The contributions of the proposed weighting method

In this section, the computational efficiency of the proposed weighting method in obtaining stable explanations is illustrated. In (Visani et al., 2020), the authors highlight the significant influence of σ in (2.6) on the stability of the explanations, which can also be observed in Fig. 4.4. The x axis represents the corresponding hyperparameters in SEGAL and SEGKL,

while the y axis represents the stability evaluations with the various hyper-parameter values. If σ is not chosen properly, the final explanations will not be stable. Consequently, they propose using Bayesian optimisation to find the optimal σ within a certain search interval. One thing to note is that finding the optimal σ is challenging. First, it is difficult to determine the search interval. Based on (2.6), the optimal σ can be expected to depend on the absolute distances between the created neighbours and the corresponding target sample. For different problems or even for different instances of the same problem, these distributions can vary significantly. Therefore, the optimal σ is expected to vary significantly across different instances. Furthermore, Fig. 4.4a shows that the stability of the explanations is highly sensitive to σ. Therefore, if the search interval is not chosen properly, achieving stable explanations becomes challenging.

To address this, in the proposed weighting method, the suitable α in (4.5) is independent of the absolute distance distributions. Therefore, the search intervals can be consistent for different problems. More importantly, as can be seen in Fig. 4.4b, the stability of the explanations is not very sensitive to α, which facilitates the optimisation method to obtain the optimal parameters.

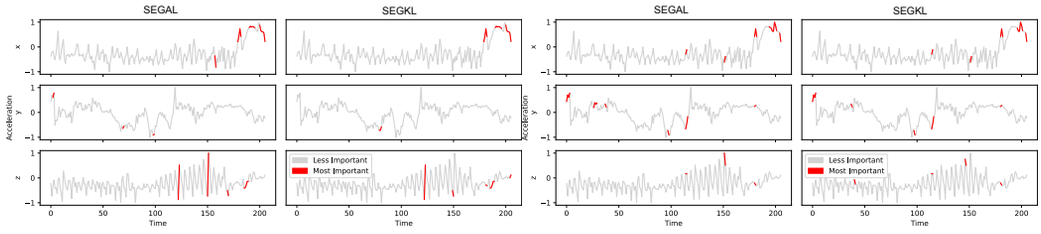
To further demonstrate the effectiveness of the proposed weighting method, the Jaccard-based stability evaluation measures and the average time of iterations carried out by the optimisation process for SEGKL and SEGAL across the adopted data sets are presented in Table 4.6. Numbers in bold indicate more stable explanations. The experimental design for these two methods is shown in Table 4.3. The stability measure indicates that the proposed weighting approach outperforms the traditional method in 11 data sets for both Transformer and LSTM classifiers. However, the p values of the Wilcoxon tests suggest that these two methods are not significantly

Table 4.6: The effectiveness of the proposed weighting method

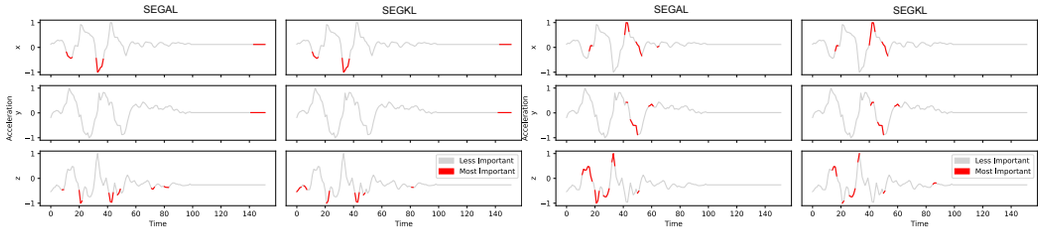
	Average of Jaccard indices				Average of Iteration Times			
	Transformer		LSTM		Transformer		LSTM	
Data set	SEGKL	SEGAL	SEGKL	SEGAL	SEGKL	SEGAL	SEGKL	SEGAL
ArticularyWordRecognition	**0.3000**	0.2711	**0.5239**	0.5047	9.00	**8.92**	**8.22**	8.34
BasicMotions	**0.7736**	0.7639	0.8752	**0.8769**	6.70	**6.25**	4.10	**3.60**
CharacterTrajectories	**0.7678**	0.7483	0.7365	**0.7429**	**6.84**	7.36	7.60	**7.42**
Epilepsy	0.8446	**0.8512**	0.8723	**0.8799**	4.86	**4.38**	5.16	**4.90**
Ering	0.7252	**0.7345**	**0.9188**	0.9056	6.50	**6.46**	**2.84**	3.10
Handwriting	0.9532	**0.9595**	0.9339	**0.9464**	2.28	**1.48**	2.16	**2.04**
JapaneseVowels	0.9090	**0.9239**	0.9169	**0.9186**	2.00	**1.70**	2.90	**1.98**
Libras	0.9364	**0.9538**	**0.9373**	0.9151	1.96	**1.50**	**1.96**	2.42
LSST	0.9416	**0.9553**	0.9624	**0.9704**	2.24	**2.12**	1.70	**1.60**
NATOPS	0.3798	**0.3859**	0.4638	**0.4659**	**9.00**	**9.00**	8.96	**8.40**
PenDigits	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.00**	**0.00**	0.02	**0.00**
RacketSports	0.9920	**0.9960**	0.9889	**1.0000**	0.80	**0.32**	0.66	**0.16**
SpokenArabicDigits	0.5690	**0.5896**	0.5849	**0.6155**	**9.00**	**9.00**	**9.00**	**9.00**
UWaveGestureLibrary	**0.4496**	0.3524	**0.6137**	0.5408	8.86	**8.82**	8.62	**7.66**
MNIST	0.8527	**0.8577**	0.8523	**0.8548**	5.98	**5.34**	5.80	**5.56**
Wilcoxon p-value	0.6377		0.2719		0.0229		0.0413	

different in terms of stability. However, with respect to search efficiency, the average iteration time, which is explained in Section 4.4.3, reveals that, by adopting the proposed weighting method, the optimisation process requires fewer iterations on 14 data sets for the Transformer and 12 data sets for the LSTM classifiers. Furthermore, the p values suggest that the difference here is significant.

The primary aim of the novel weighting method is to facilitate the optimisation process, which essentially determines how many neighbours are considered (or assigned non-zero weights) during the explanation phase. Therefore, it is important to explore whether the novel weighting method results in different explanations compared to the traditional method. To illustrate this, Fig. 4.5 shows some explanations obtained by SEGKL and SEGAL. Both of these two methods are carried out with searched optimal parameters. In Fig. 4.5, the features considered most important are coloured red. The results reveal a remarkable similarity between the explanations derived from these two weighting strategies, suggesting the near equivalence of these two methods in this case. In (2.6), when the hyperparameter σ is extremely small, only the immediate neighbours are endowed



(a) Explanations for the LSTM (b) Explanations for the Transformer



(c) Explanations for the LSTM (d) Explanations for the Transformer

Figure 4.5: Explanations provided by SEGKL and SEGAL for an instance from the Epilepsy data set ((a) and (b)) and for an instance from the Handwriting data set ((c) and (d)). The identified most important features are coloured in red. The explanations provided by these two methods are very similar.

with non-zero weights. This is equivalent to assigning a high value to α in (4.5), where only the lower-rank neighbours obtain non-zero weights. On the contrary, when σ is very large, each neighbour carrying is assigned an identical weight of '0', which is similar to assigning a very small value to α. For the local proxy interpretable model used in both SEGKL and SEGAL, the input and output are identical, and the weights are similar, since both of these weighting methods assign larger weights to closer neighbours. As a result, the final explanations provided by both SEGKL and SEGAL are expected to be similar, which are shown in Fig. 4.5.

It is intriguing to observe that the explanations for these two types of classifiers differ significantly. This suggests that if the mechanisms for classifying the same data are entirely different between the models, then the explanations may only be locally stable for each given classifier. In other words, the explanations are specific to the architecture and dynamics of the individual models, rather than providing a universally consistent

Table 4.7: Jaccard-based stability measure across explanation methods and data sets

	Transformer					LSTM				
Data set	LIME	OptiLIME	Slime	BayesLIME	SEGAL	LIME	OptiLIME	Slime	BayesLIME	SEGAL
ArticularyWordRecognition	0.2421	**0.4446**	0.2314	0.2451	0.2711	0.567	**0.5998**	0.4527	0.5681	0.5047
BasicMotions	0.4621	0.7523	0.3918	0.4542	**0.7639**	0.7227	0.8328	0.5395	0.7229	**0.8769**
CharacterTrajectories	0.4123	0.5344	0.3348	0.4142	**0.7483**	0.4791	0.5831	0.4164	0.4756	**0.7429**
Epilepsy	0.7351	0.8396	0.4758	0.7292	**0.8512**	0.8086	0.8695	0.6676	0.8117	**0.8799**
Ering	0.4102	0.5481	0.324	0.4043	**0.7345**	0.7068	0.8067	0.4862	0.6999	**0.9056**
Handwriting	0.8359	0.9224	0.4781	0.846	**0.9595**	0.9297	0.9299	0.7221	0.9336	**0.9464**
JapaneseVowels	0.7614	0.8467	0.4775	0.7491	**0.9239**	0.6687	0.8036	0.5118	0.6845	**0.9186**
Libras	0.6507	0.8053	0.4869	0.6293	**0.9538**	0.5764	0.7244	0.4865	0.5769	**0.9151**
LSST	0.5427	0.8067	0.5383	0.5539	**0.9553**	0.7311	0.8914	0.5783	0.7178	**0.9704**
NATOPS	0.1953	0.3734	0.2249	0.1941	**0.3859**	0.5538	**0.6473**	0.3789	0.5621	0.4659
PenDigits	0.9609	**1.0000**	0.8076	0.9604	**1.0000**	0.9956	**1.0000**	0.7825	0.9793	**1.0000**
RacketSports	0.5240	0.9076	0.6564	0.5307	**0.9960**	0.6558	0.9320	0.3004	0.7067	**1.0000**
SpokenArabicDigits	0.3781	0.5648	0.2540	0.3764	**0.5896**	0.5361	0.5658	0.2338	0.5297	**0.6155**
UWaveGestureLibrary	0.3885	**0.5252**	0.206	0.3903	0.3524	0.3247	0.4316	0.5269	0.3245	**0.5408**
MNIST	0.5739	0.6546	0.4112	0.5674	**0.8577**	0.6154	0.6898	0.3687	0.6058	**0.8548**
Average Ranking	3.4667	1.8333	4.7333	3.6667	**1.3000**	3.4667	1.9667	4.8000	3.3333	**1.4333**
Wilcoxon Test p-value	0.0001	0.0480	0.0000	0.0000	/	0.0004	0.0413	0.0000	0.0009	/

understanding of the features important for classification. This variability could pose challenges when interpreting model decisions for end users or when attempting to generalise these explanations across different models.

4.4.3 Comparison with benchmarking methods

The stability metrics for various explanation methods and datasets are presented in Table 4.7. The numbers in bold represent the best results for each combination of datasets and classifiers. The Wilcoxon test was performed to assess the significance of performance differences between the proposed method and other pair-based benchmarking methods. The corresponding p-values are listed in the last column of the table. Small p-values (≤ 0.05) indicate statistically significant differences between the proposed methods and the benchmarking methods adopted. Furthermore, the smaller average ranks of the proposed method suggest that it performs significantly better than Lime, Slime, and BayesLIME.

Of course, it is not enough for the explanations to be stable. For example, if an explanation method always identified the first 10 features as important, this explanation would be perfectly stable, but also likely useless.

In other words, beyond being stable, explanations should also be faithful, meaning that the features identified as important in a feature importance explanation should be really relevant to the model output being explained. However, for a black-box model, the ground truth of its explanations is lacking, which means that we do not know which features are genuinely important for its outputs. This makes evaluating the faithfulness of the explanations challenging. This is shown in the difference between the explanations generated for the different classifiers, such as those shown in Fig. 4.5.

In this work, a simple handwriting data set, MNIST, is adopted as an MTSC. On this data set, the two classifiers achieve very high prediction accuracy. Because this data set is very simple, representing a digit written on a whiteboard, we can assume that the relevant features for the classifiers are visible on (or near) the written digits. Based on this assumption, we can, to some extent, qualitatively evaluate the faithfulness of the explanations. Fig. 4.6 shows the explanations provided by the adopted methods for some selected instances from this data set. These instances are selected because the explanations provided by these methods have a similar stability evaluation. The results reveal that the relevant features identified by our method consistently align with the written digits, while other methods identify some features located in the background as important, which may not actually be relevant. These features are misidentified primarily due to the OOD problem (Hase et al., 2021). The results indicate that our method has the potential to provide not only more stable but also meaningful explanations.

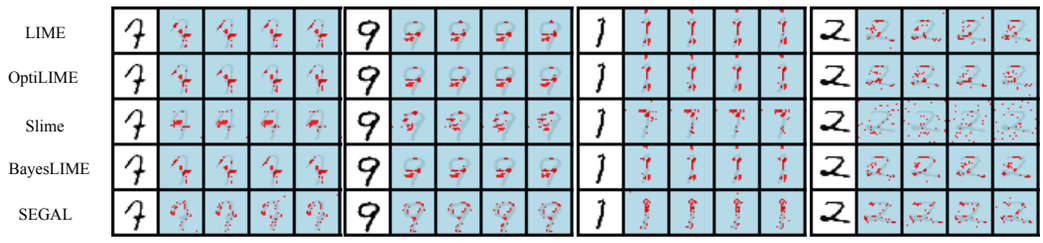


Figure 4.6: Explanations provided for selected instances from the MNIST data set, for the transformer classifier. The first column displays the original input, while the remaining columns show explanations provided by the corresponding methods that are carried out multiple times. In the explanations, important features are highlighted in red.

4.4.4 The contributions of the generative method

In this section, the contributions of the generative method to the stability of explanations are explored. For this purpose, the performance of SETKL and SEGKL is analysed. The primary difference between these two methods lies in the approach to generate neighbours, the details of which are presented in Table 4.3. The stability evaluation metrics for these two methods across the adopted data sets are shown in Table 4.8. The results show that for the transformer classifiers, SEGKL outperforms SETKL on 11 (out of a total of 15) data sets. For LSTM classifiers, SEGKL performs better on 10 data sets. The p values of the Wilcoxon tests suggest that their difference is significant. These findings imply that the integration of a generative model improves the ability of LIME to provide more stable explanations. The reasons behind this contribution from generative models are discussed below.

First, the output surfaces of the classifiers for the samples generated by the generative method are smooth. In this context, 'smooth means that the classifier output does not undergo significant changes with slight variations in the input, as illustrated in Fig. 4.7. On the contrary, the output surfaces of the classifiers on the traditionally created samples tend to be rough.

Table 4.8: Stability measures of SETKL and SEGKL across the adopted data sets

	Transformer		LSTM	
data set	SETKL	SEGKL	SETKL	SEGKL
ArticularyWordRecognition	**0.4326**	0.3000	**0.5997**	0.5239
BasicMotions	0.6793	**0.7736**	0.8567	**0.8752**
CharacterTrajectories	0.5394	**0.7678**	0.6316	**0.7365**
Epilepsy	0.8423	**0.8446**	**0.8826**	0.8723
Ering	0.489	**0.7252**	0.8441	**0.9188**
Handwriting	0.9459	**0.9532**	0.9464	**0.9915**
JapaneseVowels	0.7651	**0.909**	0.7814	**0.9169**
Libras	0.8764	**0.9364**	0.8360	**0.9373**
LSST	**0.9418**	0.9416	0.9597	**0.9624**
NATOPS	**0.4278**	0.3798	**0.5843**	0.4659
PenDigits	**1.0000**	**1.0000**	**1.0000**	**1.0000**
RacketSports	0.9604	**0.9920**	0.9809	**0.9889**
SpokenArabicDigits	0.4118	**0.5690**	0.4552	**0.6155**
UWaveGestureLibrary	0.5019	**0.4496**	0.4921	**0.6137**
MNIST	0.4947	**0.8527**	0.5597	**0.8548**
Wilcoxon p-value	0.0479		0.0413	

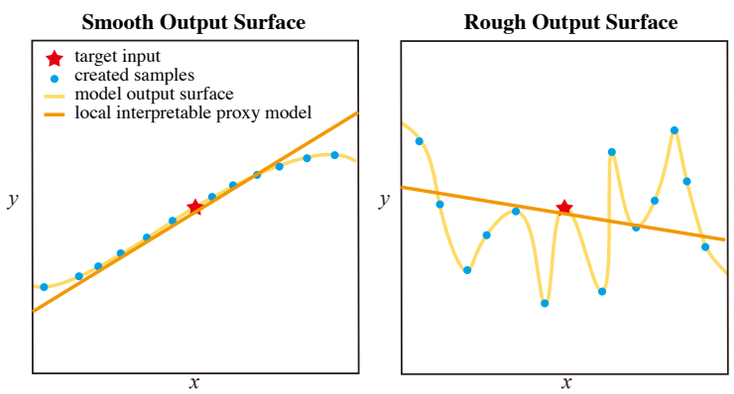


Figure 4.7: The output surfaces of the classifier on created samples.

Second, the sampling process involved in generating interpretable neighbours leads to various interpretable models, resulting in different explanations. In LIME, as shown in Fig. 4.7, if all neighbours of the target input are obtained, the final explanations will be identical. However, obtaining all neighbours becomes infeasible, especially in real-world problems with a large number of features. Since each feature of an interpretable neighbour can be '1' or '0', each target input has 2^N neighbours, where N represents the total number of features. Consequently, LIME samples a certain number of neighbours and constructs a proxy model based on these sampled

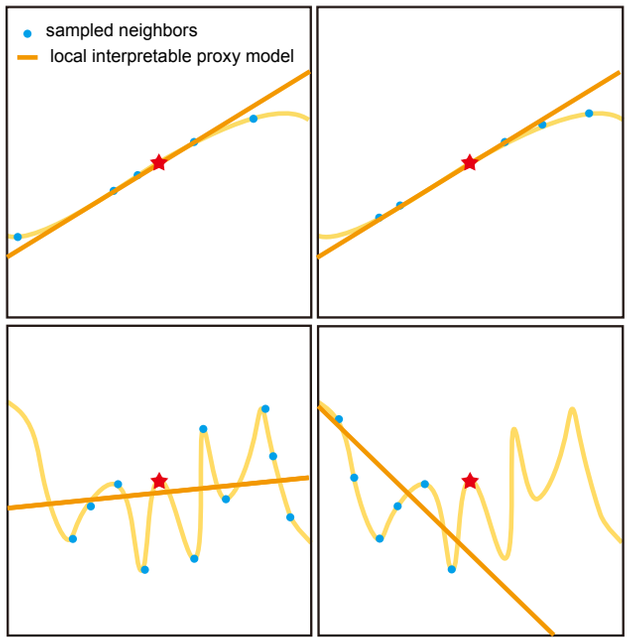


Figure 4.8: Due to the sampling process in the neighbour generation process, the rough output surface results in very different explanations (below). On the contrary, if the output surface is smooth, the final explanations tend to be stable (above).

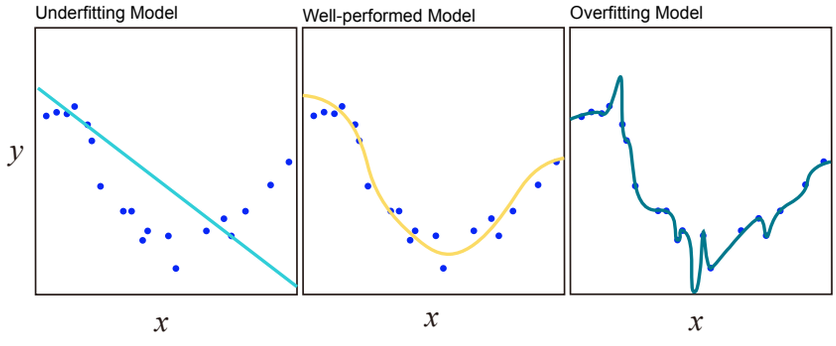


Figure 4.9: Output surfaces of the classifiers with variant performance.

instances. If the output surface of the classifiers on the sampled instances is rough, as depicted in Fig. 4.8, this sampling process may result in different final local proxy models. In contrast, if the output surface of the created neighbours is stable, as shown in Fig. 4.8 above, the sampling process will not produce significantly different local proxy models. This leads LIME to provide more stable explanations.

Therefore, the output surface of the classifier on the samples created in LIME plays an important role in the stability of the explanations. For

well-performing classifiers that generalise well on the test data sets, their local output surface will generally be smooth. This can be illustrated in Fig. 4.9. The explanations provided for these classifiers should be stable, such as those for classifiers trained on the MNIST and JapaneseVowels data sets. However, Table 4.8 shows that the explanations provided by SETKL are not stable, suggesting that the output surfaces of the classifiers on the created samples are not smooth. This is because these created samples are not drawn from the same distribution as the training data set, and the classifiers cannot generalise well on these OOD samples, resulting in rough surfaces. On the contrary, the explanations provided by SEGKL are stable, suggesting that the output surface of the classifiers is smooth on these samples created by the generative model. The reason behind this is that these samples are drawn from the distribution of the training data set, which the generative model has learnt.

In certain data sets, such as LSST, SEGKL does not outperform SETKL. This may be attributed to the limited generalisation ability of the classifier, resulting in a rough output surface similar to the output surfaces of the overfitting models shown in Fig. 4.9. In this circumstance, even if the created samples come from the same distribution as the training data set, the final explanations may still be unstable. On the other hand, for data sets like ArticularWordRecognition, the classifiers demonstrate good generalisation performance. The output surfaces of these classifiers are expected to be smooth, and the explanations provided are expected to be stable. However, for the ArticularWordRecognition problem, SEGKL does not perform well. This might be due to the fact that the proposed generative model is not powerful enough to accurately capture the distribution of the training data set. Consequently, the created samples may still be OOD. We hope that future advances in the design of generative models can

facilitate the generation of within-distribution samples.

Lastly, it is important to emphasise that achieving stable explanations is a desirable attribute for both classifiers and explanations. If the explanations are not stable, it could be due to the classifier's poor generalisation ability or the presence of OOD samples used during the explaining process. In either case, the results obtained, predictions made by models or explanations provided, would not be useful.

4.5 Summary

In this chapter, the stability issue of LIME when applied to MTSC problems was investigated, where the influence of the OOD problem induced by traditional neighbour generation methods was highlighted. We proposed to use a generative method in the LIME neighbour generation step to alleviate the OOD problem. The results indicated that by employing the proposed generative method, the explanations provided by LIME become more stable. Furthermore, the proposed adaptive weighting method further improved the computational efficiency of the explanation process.

The OOD problem is a well-recognised problem in the field of explainable artificial intelligence and significantly influences the performance of final explanations. This work is the first to thoroughly investigate its influence on the stability of explanations provided by LIME. The OOD problem often arises when samples need to be generated in the explanation process. Therefore, not only LIME but also other similar explanation methods that need to generate samples should consider this issue seriously. We hope that our findings will inspire future research to continue to address this problem.

In previous chapters, we have advanced XAI within the MTSC context by developing frameworks that provide accurate and stable feature importance explanations. However, to further enhance XAI in MTSC and broaden its impact, relying solely on feature importance explanations might not be sufficient. For many ordinary people, these explanations may not provide enough insights. In certain scenarios, they seek more intuitive explanations that can help them prevent undesirable outcomes or achieve preferred ones. Consider, for example, an early failure prediction model that predicts a machine's failure using its historical operational data, such as temperature, vibration, and rotation levels. For a machine maintainer, simply knowing that the vibration level from the past day influenced the model's prediction is not enough. They would probably want to know what specific changes in the past vibration level could prevent the predicted failure. With such detailed explanations, they can take more targeted and effective actions. In this light, counterfactual explanations emerge as a potentially more useful tool. These explanations suggest alternative scenarios where making certain changes to the current input could lead to a different outcome. Therefore, to ensure that XAI benefits a wider audience, it is crucial to develop effective methods to provide counterfactual explanations. This constitutes the next focus of this thesis.

Chapter 5

Creating Plausible Counterfactuals using Generative Models

5.1 Introduction

In the preceding chapters, we primarily focused on feature importance explanations, which are foundational and intuitive types of explanation within the XAI domain. These explanations are particularly insightful for experts and data scientists who wish to understand the specific variables that most influence model predictions with the aim of conducting feature selection to improve the model's performance or ensuring that the model's behaviour aligns with known domain knowledge. However, feature importance explanations may not be as effective for ordinary people who do not have relevant domain knowledge. They wish to obtain clearer and more straightforward explanations. In this regard, counterfactual explanations, which provide alternative scenarios or 'what-if' analyses, become particularly rel-

evant (Barredo Arrieta et al., 2020). They can help people understand how different outcomes could be achieved or what changes might lead to different model outputs, thus making the complex decision-making processes of AI models more accessible and understandable to the general public.

In the literature, some methods have been proposed to provide counterfactual explanations in the MTSC domain (Delaney et al., 2021b; Wachter et al., 2017). Existing methods usually focus on enhancing the sparsity or proximity of counterfactuals (Guidotti, 2022). Sparsity-based methods aim to reduce the number of features that should be modified to alter the outputs of classifiers. On the other hand, proximity-based approaches try to minimise the distance between the counterfactuals and the original inputs. However, the aspect of plausibility, which refers to counterfactuals being realistic and originating from the same distribution as the data set, has not received significant attention. This is the same OOD problem discussed in previous chapters, and counterfactual explanations are not exempt from this issue. If the counterfactuals do not align with the original data distribution, they are OOD, and the models' outputs on such counterfactuals are unpredictable and unreliable. Even if such counterfactuals can result in different outputs, the underlying reasons may not stem from meaningful changes in the input features but rather from the OOD problem. This scenario results in explanations that could be misleading or irrelevant for practical applications, undermining the effectiveness of counterfactual explanations, because the primary purpose of counterfactuals is not only to lead to alternative outcomes, but also to provide insights that are actionable and applicable within the real-world contexts reflected in the training data.

In previous chapters, our focus has been on feature importance explanations. Some existing research suggests that counterfactuals can be obtained

from feature importance explanations (Delaney et al., 2021b; Selvaraju et al., 2017b; Castro et al., 2009). These approaches typically involve identifying the most crucial features and then replacing these features with values from other time series instances that lead classifiers to produce different outputs (Delaney et al., 2021b). However, this blunt replacement approach may disrupt the temporal dependency inherent in time series, leading to OOD time series. The framework introduced in Chapter 3 addresses this issue by replacing crucial features with plausible values generated by a generative model. The approach proposed in this chapter aims to ensure that the counterfactuals provided are aligned with the distribution of training data, thus mitigating the risk of producing OOD counterfactuals. To do this, this approach requires an effective generative model capable of accurately estimating the distribution of training data, a task that might prove challenging in some complex real-world scenarios.

In this chapter, our objective is to develop counterfactual explanation methods that circumvent the necessity for feature importance explanations. The primary emphasis is on the plausibility of counterfactuals, ensuring that the counterfactuals provided align with the distribution of data across different applications. Throughout this thesis, we adopt post-hoc approaches to explain deep learning based classifiers, meaning that the classifiers to be explained have already been trained prior to the explanation phase. During their training phase, these classifiers process the input data sequentially, transforming the original time series into a condensed feature space. In this space, realistic data points naturally cluster tightly. Therefore, it is crucial that plausible counterfactuals also appear within these densely populated regions.

To this end, in this chapter, a novel counterfactual explanation strategy is proposed, called DensityGuide. This strategy begins by estimating the

data density distribution within the classifier's compressed feature space, utilising Gaussian Mixture Models (GMMs). Following this, a generative model is trained to create plausible counterfactuals in the densest region of this compressed feature space. However, the GMMs, which is the basis of the DensityGuide method, may not always be able to accurately capture the data distribution. To mitigate this, an alternative approach, named LatentGAN, is designed. This method avoids making specific assumptions about the data distribution and leverages the Generative Adversarial Network (GAN) framework to ensure that counterfactuals are consistent with the training data distribution in the latent feature space encoded by the classifiers. Both approaches try to create counterfactuals using generative models, with the aim of aligning created counterfactuals with the distribution of realistic time series. Experiments carried out with real-world data sets highlight the strengths of each approach in different real-world problems. The significant difference between the proposed methods and existing ones lies in their focus on aligning counterfactuals with the original data distribution. This alignment ensures that the final explanations are realistic and meaningful.

One notable advantage of counterfactual explanations over feature importance explanations is their provision of clearer, actionable insights that can guide users to take effective actions. Recently, advancements in large language models, such as ChatGPT, have opened up new possibilities for refining and communicating counterfactuals in ways that can suggest practical actions to users. To maximise the benefits of counterfactual explanations, this chapter also illustrates the use of ChatGPT to enhance their interpretability and practicality. By leveraging ChatGPT, we aim to transform these explanations into actionable guidance, bridging the gap between counterfactual explanations and real actionable guidance.

The remainder of this chapter is organised as follows. In Sections 5.2 and 5.3, the details of DensityGuide and LatentGAN are described. Section 5.4 details the experimental design used in our study. Subsequently, Section 5.5 discusses the results and analysis derived from these experiments. Section 5.6 explores how ChatGPT can be leveraged to enhance the utility of counterfactual explanations, making them even more actionable and user-friendly. The chapter concludes with Section 5.7, where we summarise our findings and their implications.

5.2 DensityGuide

In this section, we describe in detail the DensityGuide method. Figure 5.1 illustrates the workflow of this method, which includes a preliminary step and a subsequent counterfactual generation step. The fundamental principle of this approach is to ensure that the counterfactuals generated are located in the densest regions within the training data manifold (Artelt and Hammer, 2020). This strategy aims to maximise the likelihood that counterfactuals occur in reality. Therefore, this approach begins with an estimate of the density distribution of the training data, a process carried out in the preliminary step. Following this, the counterfactual generation step involves optimising a generative model. This optimisation objective is to maximise the probability density of the counterfactuals, thereby aligning them closely with the most likely scenarios in the training data. The details of these two substeps are described below.

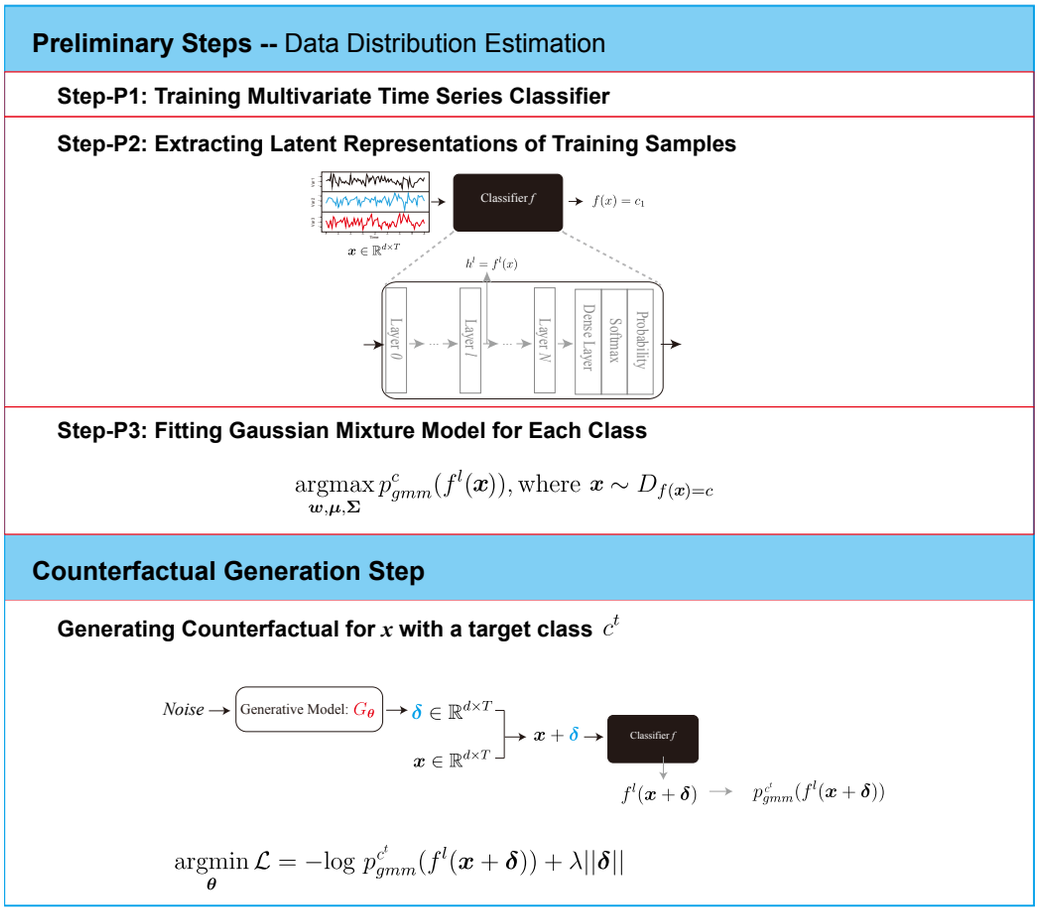


Figure 5.1: The workflow of the proposed method.

5.2.1 Preliminary Steps

The preliminary steps aim to estimate the density distribution of the training samples. Because the original inputs are multivariate time series, which are high-dimensional and have complex dependency between features, estimating their data distribution directly is much harder than that of tabular data. Therefore, the data distributions of the training samples are estimated using their latent representations encoded by the classifier, of which the details are explained as follows.

In this thesis, the objective is to explain MTSC using a post-hoc approach. Therefore, the initial step involves fitting a classifier. In the subsequent step, the latent representations of the training samples are extracted using the trained classifier. In this thesis, we focus on deep learning-based clas-

sifiers. Therefore, classifiers, as illustrated in Fig. 5.1, are usually made up of multiple layers. These layers sequentially process the input data, transforming it from its original feature space into a series of latent representations and ultimately into the final output space. This final output space represents the probabilities of the predicted classes. A certain latent representation of the input can be denoted as $f^l(\boldsymbol{x})$, where l specifies that this latent representation is the output of the l-th inner layer. In our approach, we extract the latent representations from the layer immediately preceding the final dense layer that are followed by a softmax layer to generate the probabilities of the different classes.

In the next step, GMMs are fitted to the training samples using the extracted latent representations. The choice of GMMs is motivated by several factors. First, GMMs are adept at estimating a variety of data distributions because of their composition of multiple independent Gaussian components. This ability makes them particularly versatile for modelling complex data patterns. Second, GMMs are characterised by their clear analytical representations, which can be calculated straightforwardly and incorporated into the construction of the loss function necessary to generate counterfactuals in the subsequent step. Specifically, for each predicted label, a distinct Gaussian model is fitted to capture the distribution of the samples corresponding to its respective label, which can be mathematically denoted as follows.

$$\underset{\boldsymbol{w},\boldsymbol{\mu},\boldsymbol{\Sigma}}{\operatorname{argmax}} \; p^c_{gmm}(f^l(\boldsymbol{x})), \text{where } f(\boldsymbol{x}) = c \tag{5.1}$$

where $\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ are the weights, means, and the covariance of the corresponding p^c_{gmm} fitted for the samples predicted as class c.

The use of latent representations to estimate data distributions offers another advantage beyond reducing the difficulty of estimating data distribu-

tions. In the training phase, classifiers are trained to convert the original inputs into a more compact and informative latent feature space. In this feature space, training samples predicted to have the same labels tend to be close to each other. Therefore, if a new input comes from the same distribution of the training data, its latent representations should also be located in the density area in the latent features of the classifier. On the contrary, estimating the data distribution directly from the original input, particularly in time series data, often involves encoding the input into a distinct feature space. This new space can be vastly different from the latent space used by the trained classifier, leading to discrepancies. For instance, samples that appear similar in the classifier's space might be deemed different in the separately trained encoder's space. In the context of counterfactual generation, our aim is to produce samples that closely resemble those in the training data set. Estimating the data distribution within the latent feature space of the classifier is crucial for this purpose. If the estimation is conducted directly in a different latent space, it risks deviating from the perspective of the classifier, which can lead to less accurate or relevant counterfactuals. Therefore, leveraging the latent feature space of the classifier is a strategic choice to closely align the counterfactual generation with the training data.

5.2.2 Counterfactual Generation Step

After obtaining the density distribution of the training data, counterfactual explanations can be generated from the densest region. The current work focuses on deep learning-based classifiers of which the gradient is accessible; thus, the objective of generating counterfactual explanations is treated as an optimisation problem and resolve it through gradient-based methods.

Specifically, details are shown in Fig. 5.1.

The classifier predicts that $\boldsymbol{x}$ belongs to class c^{ori}. We are interested, for example, in how the $\boldsymbol{x}$ change can cause the classifier to produce another label, denoted c^t. To answer this question, we propose to adopt a generative model that takes pure random noise as input and produces a mask $\boldsymbol{\delta}$, which is the same size as $\boldsymbol{x}$. The counterfactual is constructed by adding the generated mask over the original input, denoted as $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$.

Our objectives in counterfactual generation are two-sided. First, we wish that the counterfactuals obtained are located in the densest region of the samples predicted belonging to class c^t. Mathematically, this objective can be formulated as maximising the log-likelihood of the counterfactuals using the corresponding fitted GMM $p_{gmm}^{c^t}$. The second objective is that we wish that the changes made on $\boldsymbol{x}$ are as small as possible, which is equivalent to obtaining a sparse mask. Therefore, the training objective of the generative model can be represented as follows:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L} = -\log\, p_{gmm}^{c^t}(f^l(\boldsymbol{x} + \boldsymbol{\delta})) + \lambda||\boldsymbol{\delta}|| \tag{5.2}$$

where $p_{gmm}^{c^t}$ represents the obtained GMM model for the samples predicted belonging to c^t; λ is a hyperparameter that controls the trade-off between the sparsity of the mask and the log-likelihood of the obtained counterfactuals. In the optimisation stage, we are going to update the parameters $\boldsymbol{\theta}$ within the generative model through gradient-based optimisation method.

The choice of λ. The selection of the parameter λ is crucial for the quality of the final counterfactuals. If λ is too large, the modifications applied to $\boldsymbol{x}$ will be minimal. Consequently, the resulting counterfactual may not be classified as belonging to c^t or may not fall within the densest region of the

distribution. On the other hand, a small λ value may lead to a non-sparse mask, making the counterfactual less interpretable for users.

To address this difficulty, we have proposed an automatic adjustment method for λ, of which the details are shown in Algorithm 2. Initially, λ is set to a small value (1.0 in this work), and the generative model is optimised to maximise the likelihood of counterfactuals. Due to the nature of the GMMs adopted in this work, there will be a point where the likelihood of $\boldsymbol{x}'$ cannot be further increased, leading to a plateau. Therefore, in Algorithm 2 (lines 14 to 17), operations are carried out to judge whether the plateau is reached. This is achieved by assessing whether the maximum of the last 100 log-likelihood values recorded in LogProbSeq exceeds the minimum of these by a predefined margin. If the plateau is reached, the average of these last 100 log-likelihood values then defines the corresponding plateau level. Once this plateau is reached, λ is adjusted based on the current likelihood. If the current likelihood is smaller than the plateau level, this suggests space for further modifications to the input. Consequently, λ can be increased. On the contrary, if the current likelihood is lower than the plateau level, indicating excessive modifications, λ should be reduced accordingly. Using this proposed adjustment approach, the sparsity of the mask obtained $\boldsymbol{\delta}$ can achieve the highest while maintaining the likelihood of counterfactuals within the plateau level. The optimisation stops when the sparsity of the obtained mask no longer improves over a certain number of iterations. This approach ensures that counterfactual explanations achieve both high likelihood and optimal sparsity.

Generative model architecture. The architecture of our generative model, illustrated in Fig. 5.2, is based primarily on LSTM (long-short-term memory). The generative model receives pure noise sampled independently from a standard normal distribution. This sampled noise is processed in

Algorithm 2 Generative Process for Counterfactuals

Start Generative Process:
Initialise:
  LogProbReachPlateau ← False
  LogProbPlateauLevel ← Inf
  $\lambda \leftarrow 1.0$
  LogProbSeq ← [ ]
Begin Loop:
while optimisation not converged **do**
    Sampling $\boldsymbol{\epsilon} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{\delta} \leftarrow G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$
    $\boldsymbol{x'} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$.
    logprob ← $p^{c^t}_{gmm}(\boldsymbol{x'})$
    LogProbSeq.append(logprob)
    if Δlogprob < threshold **then**
        LogProbReachPlateau ← True
        LogProbPlateauLevel ← Calculate Plateau Level
    end if
    if LogProbReachPlateau **then**
        if logprob > LogProbPlateauLevel **then**
            $\lambda \leftarrow \min(\lambda \times 1.05, 1e8)$
        else
            $\lambda \leftarrow \max(\lambda \times 0.95, 1.0)$
        end if
    end if
    $\mathcal{L} \leftarrow -\log\, p^{c^t}_{gmm}(f^l(\boldsymbol{x} + \boldsymbol{\delta})) + \lambda||\boldsymbol{\delta}||$
    Update $\boldsymbol{\theta}$ in $G_{\boldsymbol{\theta}}$ using Adam optimisation method.
end while

sequence by the LSTM cells, and the LSTM outputs are further processed by a dense layer, which then passes through a hyperbolic tangent (Tanh) activation function, scaled by a factor of 2. This adjustment ensures that the output values change from -2 to 2, aligning with the preprocessing normalisation of the training samples, which are scaled to fall between -1 and 1. Consequently, this scaling guarantees that the maximal alteration applied by the mask to the input remains within the bounds of -2 and 2. In particular, the parameters of the dense layer are shared across all time steps, which aims to ensure a uniform transformation of the LSTM output and maintain the temporal coherence of the generated sequence.

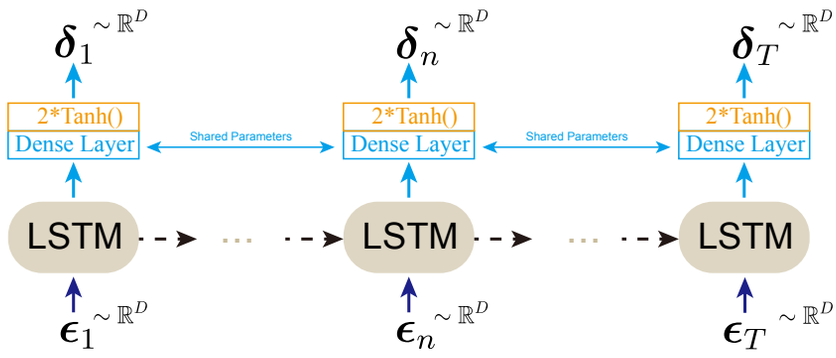


Figure 5.2: The architecture of the adopted generative model

5.3 LatentGAN

In our first proposed method, we hypothesise that the density distribution of the training samples in the latent features space follows a mixture of Gaussian distributions. Although GMMs are adept at capturing complex data distributions, they may not always be suitable, particularly in scenarios where the data deviates from mixture Gaussian distribution patterns. This discrepancy can hinder the generation of high-quality counterfactuals. To overcome this limitation, our second approach uses the GANs architecture. The key advantage of GANs lies in their ability to ensure that both the generated counterfactuals and the real time series originate from the same distribution without making explicit assumptions about the type of distribution, offering a more flexible and robust solution for counterfactual generation.

The GAN framework is also used in the development of counterfactual explanation methods (Lang et al., 2023b). In the GAN architecture, the generator network aims to create synthetic data that mimic the real data set, while the discriminator network aims to distinguish the generated (fake) data from the actual (real) data. These two networks are optimised iteratively, where the generator strives to become increasingly proficient at producing realistic data, and the discriminator becomes better at detecting

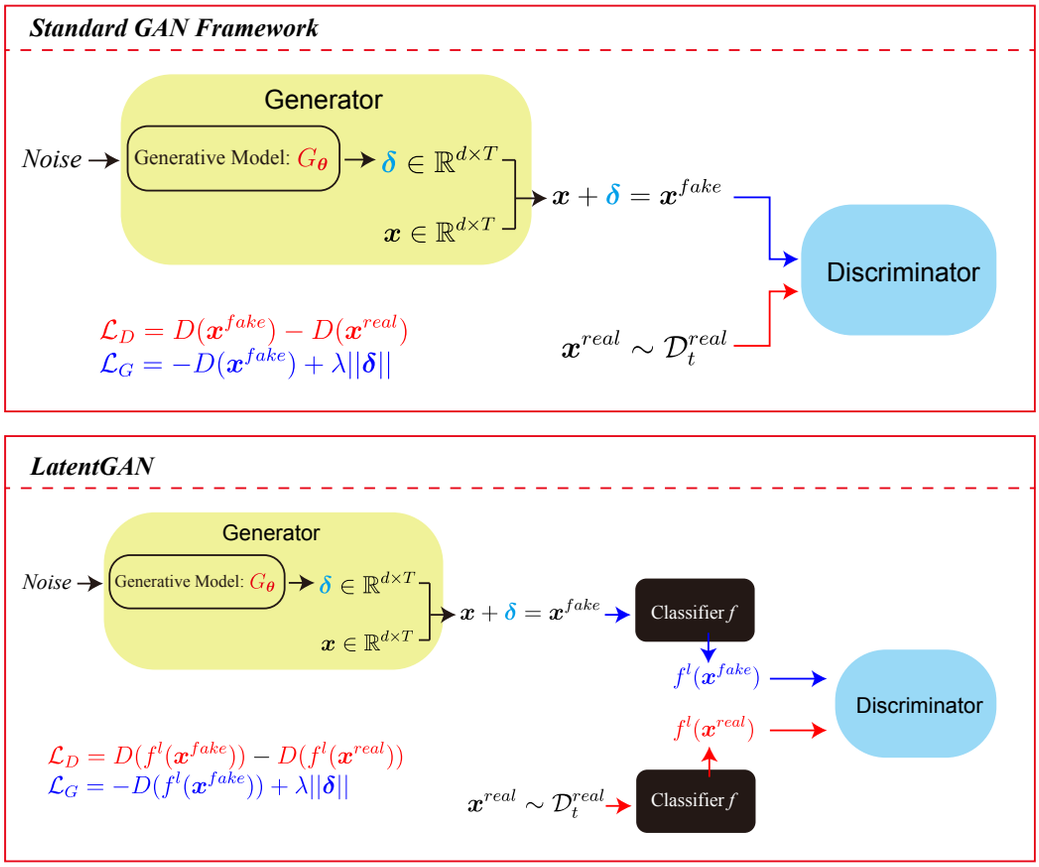


Figure 5.3: Standard GAN framework and the LatentGAN framework used in this work.

the differences.

In the context of counterfactual explanations, the GAN is adapted to suit this specific task. The generator is optimised to produce counterfactuals that are indistinguishable from realistic data classified as belonging to a particular class. This concept is illustrated in the standard GAN framework shown in Fig. 5.3. Here, $\boldsymbol{x}^{real}$ represents the real time series that the classifier predicts as belonging to class t, while $\boldsymbol{x}^{fake}$ denotes the synthetic time series generated by the GAN. In the GAN framework, the term $\lambda||\boldsymbol{\delta}||$ in generator loss is used to force the generator to produce sparse counterfactual masks, thus improving the interpretability of the final explanations. The generator and discriminator are iteratively optimised to produce counterfactuals that are indiscernible from real time series associated with

a certain predicted class, thus enhancing the realism and relevance of the generated counterfactuals.

However, as discussed in Section 2.5.2, adopting the standard GAN framework presents significant challenges. These difficulties arise from various factors, such as the training imbalance between the generator and the discriminator, which is exacerbated by the complexity inherent in time series data. The generator, trained to create realistic time series, often faces greater difficulties, while the discriminator can more easily distinguish between real and fake data. Although fine-tuning the architecture of the generator and discriminator can mitigate these issues to some extent, designing a universal architecture that is suitable for various data sets remains a significant challenge, which impedes the use of GAN to generate counterfactual explanations.

Consequently, instead of using the standard GAN framework, a LatentGAN approach is developed, as shown in Fig. 5.3. In the LatentGAN approach, the generator maintains the same architecture as the standard GAN, and a key difference lies in the input to the discriminator. In this approach, the discriminator receives latent representations of time series $f^l(\boldsymbol{x}^{fake})$, which are encoded by the classifier as described in Section 5.2.1. Consequently, the objective of the generator is to produce counterfactuals that are indistinguishable in the latent feature space encoded by the classifier. This ensures that the counterfactuals generated align with the distribution perceived by the classifier.

In our practice, training this LatentGAN has presented fewer difficulties. The reason might be that the classifier's involvement aids the generator in focusing on making necessary alterations only to the relevant features, while leaving unaffected those features that do not affect the model's per-

formance. As a result, the complexity of the task for the generator is substantially reduced, enhancing the overall efficiency and effectiveness of the LatentGAN in generating plausible counterfactuals.

5.4 Experimental Design

In this section, the experimental design for this work is provided. First, the adopted data set and the corresponding preprocessing approaches are described. Then, comparable methods will be introduced, of which the performance is compared with the proposed methods. Finally, the evaluation metrics used in this work are introduced.

5.4.1 Data sets and Classifier

To evaluate the performance of our proposed framework, we use the same data sets as those detailed in Section 4.3.1. This selection includes 14 data sets from the UEA archive (Bagnall et al., 2018) and the MNIST handwriting data set (Lecun et al., 1998). For consistency, we apply identical data preprocessing techniques as outlined in the preceding chapter. In addition, the same LSTM-based classifier is used.

In this work, we also use the Microsoft Azure Predictive Maintenance data set[1]. This data set comprises hourly averages of the voltage, rotation, pressure, and vibration measurements collected from 100 machines. We focus on predicting failures using this data set. Specifically, we used the 48-hour historical time series data of voltage, rotation, pressure, and vibration as input to our classifier. The same LSTM-based classifier as that described

[1]This data set is obtained from https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance/data

in Section 4.3.1 is implemented and trained on this data set. This classifier is designed to predict the probability of machine failure in the next 24 hours. The selection of the Microsoft Azure Predictive Maintenance data set for this chapter is motivated by the clear physical interpretations of its time series data, such as voltage, rotation, pressure, and vibration measurements. On the contrary, the data sets in the UEA archive contain time series whose meanings are less clear. Using a data set with well-defined physical attributes allows us to more effectively demonstrate how counterfactual explanations can offer actionable guidance, showcasing their practical benefits in scenarios with clear real-world relevance.

5.4.2 Benchmarking Methods

There have some method proposed to provide counterfactuals explanations for models. Some of them are initially designed for time series models, while some methods are initially designed for tabular data. In this work, we introduce some classic approaches and compare them with our proposed method.

WACH (Wachter et al., 2017): The WACH method, introduced by Wachter et al., stands out as one of the most well-known counterfactual explanation techniques. WACH generates explanations by optimising the objective defined in (2.8). In this optimisation objective, the parameter λ plays a crucial role, balancing the closeness of the classifier's output to the desired outcome against the similarity of the counterfactuals to the original input. Following the Wachter et al. guidelines (Wachter et al., 2017), our approach seeks to maximise λ while minimising the difference between the classifier output and the desired result. Specifically, optimisation begins with a small initial value of $\lambda = 1e - 4$, focussing first on increasing the

probability of the target class produced by the classifier. This process continues until the predicted probability reaches a predefined threshold, set here at 0.999. Once this threshold is achieved, λ is adjusted adaptively, similar as that outlined in Algorithm 2 (lines 19 to 23). Specifically, λ decreases if the predicted probability is below the threshold and increases otherwise. This strategy aims to yield sparse counterfactuals while ensuring that the classifier's output reaches our desired outcome. In this work, we use the MSE method to measure the distance between the original input and the counterfactual.

NaiveGuide (Delaney et al., 2021b): This method was initially developed for univariate time series classifiers but can be readily adapted to multivariate contexts. The process begins by identifying the most distinctive segments of the input using attribution methods. Then, these identified segments are replaced with those of their nearest neighbours. In the original study (Delaney et al., 2021b), a convolutional neural network classifier was used, which facilitated the use of Class Activation Maps (CAM) (Zhou et al., 2016) to pinpoint the most significant subsequences. In our current work, we use classifiers based on recurrent neural networks. Consequently, we adopt Kernel SHAP (Lundberg and Lee, 2017) to identify the most distinctive segments. The SHAP kernel uses the LIME framework to compute Shapley values, which offers a more efficient way to derive these values than direct computation. After calculating the Shapley values for the input time series, we perform a smoothing operation. This step is designed to eliminate the effects of noise, ensuring that the most distinctive time points remain continuous over time. Following this, we select the closest samples with the desired predicted labels from the training data set. We then gradually replace the most distinctive time points of the current series with those of the closest sample until the classifier produces the desired la-

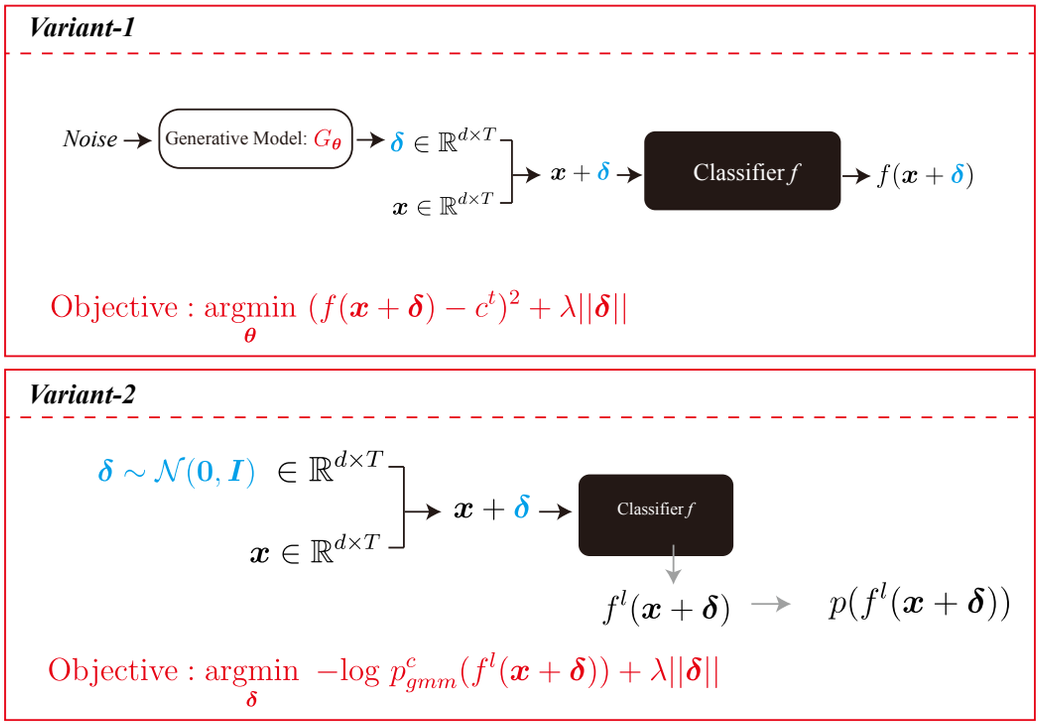


Figure 5.4: Investigated variants of the proposed method

bel. In this work, we use the MSE method to measure the distance between the time series to find their closest counterfactual sample.

Beyond the comparison methods mentioned above, we have also developed several variants of the DensityGuide method to assess the contributions of specific components designed in this approach. The architectures of these variants are illustrated in Fig. 5.4.

DensityGuide-V1: This version focuses solely on the output of the classifier, with the aim of maximising the predicted probability of the desired class. It does not consider the density probability of the resulting counterfactual. The purpose of this variant is to evaluate the necessity of ensuring that the counterfactuals have the highest possible density estimate.

DensityGuide-V2: Unlike the original method, where parameters in the generative model are optimised, this variant introduces and optimises a randomly initialised mask directly. This approach is designed to explore

the effect of the generative model in the counterfactual generation process.

5.4.3 Evaluation Metrics

The properties that a good counterfactual should meet have been introduced in Section 2.5.1, which comprises the quantitative evaluation metrics used in this chapter.

Proximity: This metric assesses the closeness of the counterfactuals to the original input time series. Various measures can be used to evaluate the similarity between two time series. In our study, we evaluate the proximity of the counterfactuals provided using Dynamic Time Warping (DTW).

Sparsity: This metric evaluates the extent of modifications in the original time series necessary to generate the counterfactuals. In our approach, changes are implemented by adding a generated mask to the original time series. Although each time step might undergo modifications, only a very limited portion of these changes are significant. Therefore, we count only those time steps where modifications exceed a noticeable threshold. Given that all time series data in our study are scaled to the range [-1, 1], we set this threshold at 0.01. This implies that any absolute changes greater than 0.01 at the time steps are considered noticeable.

Plausibility: This metric evaluates the likelihood that counterfactuals originate from the same distribution as the training data set. This assessment is vital to ensure their relevance, realism, and comprehensibility. Directly measuring the probability density of generated counterfactuals poses a challenge because of the unknown nature of the true distribution. In our study, we adopt metrics commonly used in assessing the performance of generative models (Dhariwal and Nichol, 2021), notably the Fréchet Inception

Distance (FID). The aim of FID in this context is to quantitatively measure how well the distribution of generated counterfactuals approximates the distribution of original data, based on deep features extracted by a model trained on a broad dataset. Therefore, this metric is especially suitable for our context, which involves generating counterfactual explanations that must closely mirror the real data distribution to ensure both their realism and relevance. In our work we extract latent features from both the real time series and their corresponding counterfactuals from the trained classifier. Subsequently, the mean and covariance of these feature vectors are calculated. Under the commonly held assumption in the FID application that these features conform to a multivariate Gaussian distribution, we calculate the Fréchet distance between these distributions. A smaller Fréchet distance implies greater similarity, indicating a closer alignment between the counterfactuals and the original data distribution.

Stability: Stability is a crucial attribute of explanations, especially for counterfactuals upon which users may base their actions. Unstable explanations could lead to confusion as users struggle to determine which explanation to trust and what actions to take. To assess the stability, we measure the consistency of the explanations. This is achieved by generating explanations multiple times under the same conditions and then calculating the variance between them. The lower variance indicates higher stability, which implies that explanations are reliable and can be trusted for the decision-making process.

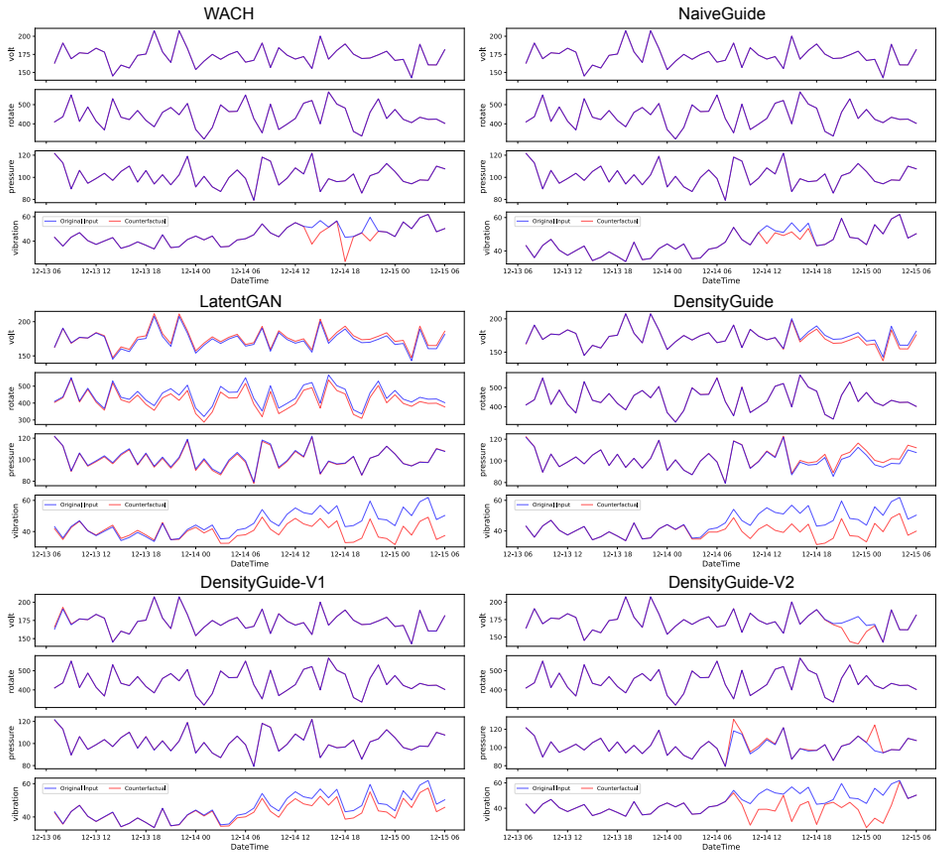


Figure 5.5: Counterfactual explanations for the classifier trained on the Microsoft Azure Predictive Maintenance data set. The original time series, shown in blue, leads to a prediction of machine failure in the next 24 hours. On the contrary, for the counterfactuals provided, coloured red, the classifier predicts that the failure will not happen in the next 24 hours.

5.5 Results and analysis

In this section, the results of the experiments are analysed in detail. In Section 5.5.1, we begin with a qualitative assessment of the counterfactual explanations provided by the methods implemented. Following this, we conducted a quantitative evaluation that measures the similarity, sparsity, feasibility, and stability of the explanations in Section 5.5.2 to 5.5.4.

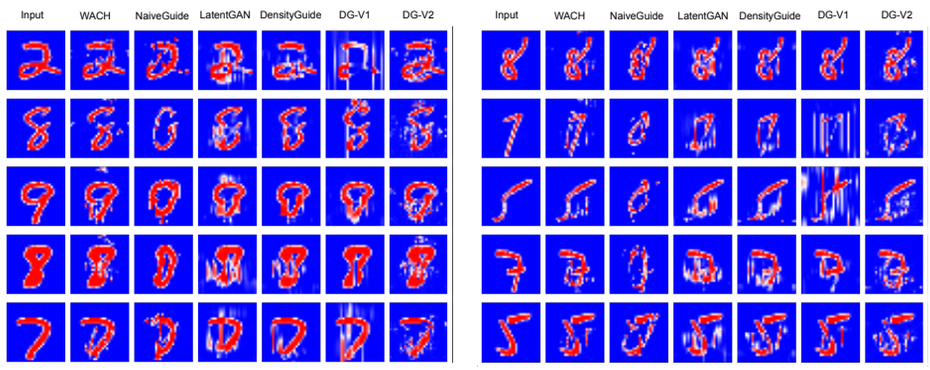


Figure 5.6: Counterfactual explanations provided by the adopted methods. The label predicted by the classifier for these counterfactuals is label "0".

5.5.1 Qualitatively Evaluation

The counterfactual explanations generated by the adopted method are shown in Fig. 5.5 and Fig. 5.6. The results indicate that the counterfactuals provided by WACH and DensityGuide-V2 differ from the original time series at separate time steps. This suggests that these methods try to alter the values in specific time steps to prompt the classifier to produce a particular output, which might break the temporal dependency within the original input. In contrast, the counterfactuals generated by other methods tend to differ from the original time series in continuous time steps. Specifically, NaiveGuide are designed to identify continuous distinctive segments for replacement, while LatentGAN, DensityGuide, and DensityGuide-V1 overlay a LSTM-generated mask on the original series, leading to continuous difference in time between counterfactuals and the original time series.

When comparing the DensityGuide-V1 method with both the DensityGuide and the LatentGAN method, extra differences between counterfactuals and the original input time series are observed. In the DensityGuide, these extra differences are particularly notable in terms of voltage and pressure levels. Meanwhile, in the LatentGAN method, extra differences are evident in voltage and rotation measurements. Although all counterfactuals

Table 5.1: Proximity evaluation of the explanations provided by the adopted methods across the data sets.

data sets	Proximity (the lower the better)					
	WACH	NaiveGuide	LatentGAN	DensityGuide	DensityGuide-V1	DensityGuide-V2
ArticularyWordRecognition	**10.5293**	20.0463	13.8009	20.5685	26.7308	18.9135
BasicMotions	**4.6111**	9.6063	6.1864	8.0649	7.7164	8.5026
CharacterTrajectories	10.6078	8.7463	**6.5735**	10.7624	9.1408	14.2606
Epilepsy	12.3536	10.4183	**7.2021**	16.0025	12.5248	13.2538
ERing	**4.2136**	8.0621	4.3260	8.0590	7.2547	8.0250
Handwriting	**4.2963**	7.6958	4.6240	8.8517	8.1265	8.6670
JapaneseVowels	**4.1099**	7.4337	6.3932	9.7527	5.6452	8.9386
Libras	5.4084	**4.2025**	6.7054	10.5639	5.6985	8.9396
LSST	5.6255	**5.2105**	7.4982	9.9473	7.3489	8.7021
NATOPS	**8.0676**	15.8449	12.1698	14.3732	12.0756	12.9017
PenDigits	**1.4921**	1.6729	2.8709	2.5375	1.6915	2.3017
RacketSports	**2.0907**	2.2913	3.3500	2.3883	2.3024	2.3741
SpokenArabicDigits	**7.4783**	13.8130	13.9141	14.7222	15.2104	17.5544
UWaveGestureLibrary	**7.6352**	15.2986	13.9162	19.2548	20.7006	10.0828
MNIST	**6.9102**	11.4213	9.9655	8.4163	10.0509	8.4782
PdM	**6.2404**	16.2685	14.6299	11.5064	10.0895	11.0797

provided by these three methods differ from the original time series in vibration level, the differences for LatentGAN and DensityGuide are more significant. These findings imply that both the DensityGuide and LatentGAN necessitate more extensive modifications to the original time series to fulfil their respective requirements. The DensityGuide-V1 method only tries to adjust the classifier for a specific class prediction with a desired probability. However, the DensityGuide tries to align the counterfactuals' latent representations with the densest regions of the training data manifold, and the LatentGAN focuses on ensuring that its counterfactual latent features are indistinguishable from those of the training data. The latter two criteria are more stringent than that of the DensityGuide-V1 method, necessitating more substantial modifications in the original time series.

5.5.2 Proximity and Sparsity Evaluation

In this section, the proximity and sparsity of the counterfactuals generated by the adopted methods are analysed, and the corresponding evaluation metrics are detailed in Table 5.1 and Table 5.2. The findings indicate that the WACH method generally results in the smallest changes to the original

Table 5.2: Sparisity evaluation of the explanations provided by the adopted methods across the data sets.

data sets	Sparisity (the lower the better)					
	WACH	NaiveGuide	LatentGAN	DensityGuide	DensityGuide-V1	DensityGuide-V2
ArticularyWordRecognition	**147.83**	774.79	705.56	1096.27	999.41	584.56
BasicMotions	**54.68**	279.05	313.38	346.26	495.95	244.20
CharacterTrajectories	**262.31**	327.07	380.91	451.88	479.13	442.00
Epilepsy	**279.05**	434.87	525.61	535.26	479.36	521.74
ERing	**31.00**	98.40	100.89	147.40	123.80	95.80
Handwriting	**45.55**	119.82	148.20	170.87	161.82	102.85
JapaneseVowels	**27.90**	152.08	180.35	179.27	87.40	156.63
Libras	47.00	**27.00**	81.75	87.50	54.00	89.00
LSST	55.64	**38.48**	120.72	126.85	59.40	116.13
NATOPS	**91.40**	588.30	666.50	854.30	762.40	416.60
PenDigits	**5.43**	9.33	14.71	11.88	5.86	11.80
RacketSports	**5.17**	9.43	19.95	17.56	7.75	14.94
SpokenArabicDigits	**561.11**	753.25	695.56	998.33	666.71	655.14
UWaveGestureLibrary	**53.50**	398.86	922.75	925.83	929.50	251.00
MNIST	**58.02**	105.19	271.72	152.86	217.33	300.26
PdM	**77.08**	151.08	402.88	218.35	209.66	371.84

time series when generating counterfactuals. This is evidenced by its lower proximity measures in 12 of 16 data sets and the lower sparsity measures in 14 of them. Such outcomes are consistent with WACH's strategy, which aims to modify the classifier output by altering the fewest possible features, disregarding the distribution of data sets. However, this approach has a risk: The counterfactuals it generates may not come from the same distribution of training data. The changes leading to different outputs are not due to meaningful alterations in the inputs, but are attributed to the OOD issue. Consequently, these counterfactuals might not accurately represent realistic and actionable scenarios, potentially leading to explanations that are not practically useful. In contrast to WACH, DensityGuide-V2 takes into account the distribution of data, resulting in more significant modifications to the original time series when generating counterfactuals. This is evidenced by its higher proximity and sparsity measurements in all data sets.

Compared to DensityGuide-V2, DensityGuide adopts a generative model based on LSTM cells, which constrains counterfactuals different from the original time series in a continuous time step. To locate the counterfactual in the densest regions and maintain the changes continuous in time at the

same time, DensityGuide needs to make more changes on the original time series than DensityGuide-V2, which is reflected in the higher proximity measurements (12 of the 16 data sets) and greater sparsity measurements (13 of the 16 data sets).

Compared to DensityGuide, the counterfactuals provided by LatentGAN make fewer changes than the original time series, which is reflected in the higher proximity measurements (all the 16 data sets) and greater sparsity measurements (13 of the 16 data sets). This is because DensityGuide tries to locate the counterfactuals in the densest regions of the data manifolds, which are described by the fitted GMMs. However, LatentGAN only maintains that the counterfactuals are not distinguishable by the classifier, which might be considered as a loose constraint on the changes made on the original time series.

Compared with DensityGuide, the counterfactuals provided by DensityGuide-V1 make fewer changes to the original input, as evidenced by its smaller proximity measurements (11 of the 16 data sets) and smaller sparsity measurements (11 of the 16 data sets). This could be explained that DensityGuide-V1 only forces the classifier to make the corresponding predictions, but does not constrain that it need to be located in the densest regions of the data manifold. This could be considered as looser constraint.

5.5.3 Plausible Evaluation

In this section, we aim to assess the plausibility of counterfactual explanations produced by the adopted methods. The FID scores are presented in Table 5.3, which measures how closely the counterfactuals provided align with the original training data distribution. The results reveal that the

Table 5.3: Plausibility evaluation of the explanations provided by the adopted methods across the data sets.

data sets	Plausibility (FID scores) (the lower the better)					
	WACH	NaiveGuide	LatentGAN	DensityGuide	DensityGuide-V1	DensityGuide-V2
ArticularyWordRecognition	10.5173	17.4096	**7.3682**	6.2248	12.1058	8.0211
BasicMotions	45.2987	10.2264	4.6554	**4.3153**	17.2505	2.1961
CharacterTrajectories	26.7283	20.4959	**14.8861**	17.7778	53.7935	17.7359
Epilepsy	32.4024	26.7948	19.8340	**14.7004**	25.7584	15.0351
ERing	31.5238	21.7975	11.5931	**11.1213**	13.5479	14.3911
Handwriting	44.5989	30.6985	14.7712	**12.5740**	14.6117	13.4417
JapaneseVowels	5.5745	5.0735	**1.8399**	4.9583	4.8500	3.6836
Libras	10.1284	10.4941	**4.9808**	5.2382	10.6462	10.2583
LSST	13.4618	12.4550	6.1111	**6.0459**	9.7074	10.9258
NATOPS	25.2495	24.6456	**12.8088**	14.6802	15.0169	15.4695
PenDigits	2.7555	3.2992	**2.2439**	2.2660	2.8852	3.8367
RacketSports	3.8984	4.6464	2.8590	**2.5620**	3.1118	3.5836
SpokenArabicDigits	16.7902	27.8419	**13.8540**	15.1065	17.5892	16.3648
UWaveGestureLibrary	12.4342	19.9207	11.8067	**10.0830**	10.4584	12.3771
MNIST	1.1116	4.6246	**0.2496**	0.3151	1.0792	0.5464
PdM	1.3652	4.9247	**0.3134**	0.4531	1.2347	0.6727

counterfactuals from NaiveGuide and WACH are associated with higher FID scores compared to other methods. This suggests a notable deviation of these counterfactuals from the training data distribution, which could indicate less plausible explanations. On the contrary, the DensityGuide and LatentGAN approaches show lower FID scores, reflecting their greater efficacy in addressing the OOD problem and providing plausible counterfactuals.

A specific comparison between DensityGuide and its variant, DensityGuide-V2, reveals that the latter exhibits higher FID scores in 13 of 16 data sets. The primary distinction between these two methods lies in that DensityGuide uses a generative model based on LSTM cells to ensure that the changes made on the input are continuous in the time step. The lower FID scores associated with DensityGuide underscore the effectiveness of this generative model in creating more plausible counterfactuals. This advantage likely stems from the model's ability to preserve the temporal dependencies inherent in time series data, ensuring that the obtained counterfactuals remain consistent with the original data distribution. Furthermore, when comparing DensityGuide with DensityGuide-V1, the latter shows higher FID scores in 14 of the 16 data sets. This outcome indicates

that focusing solely on the final output probability during counterfactual generation may risk generating OOD counterfactuals.

5.5.4 Stability Evaluation

For counterfactual explanations to be effectively used in decision-making processes, their stability and consistency are crucial. Consistency and stability mean that when the same counterfactual explaining process is applied to the input requiring explanation, the resulting counterfactuals are identical (or at least similar). In other words, the final counterfactuals eliminate the impact of stochastic components, such as randomly initialised generative models. This enhances user trust in the final explanations. Without these qualities, it becomes difficult for users to determine which explanations to trust and act upon. This section aims to investigate the stability of the explanations provided by the methods we have adopted. The quantitative metrics for this evaluation are presented in Table 5.4.

Among the methods evaluated, the NaiveGuide method, which provides deterministic counterfactuals by identifying and replacing the most important features with those of the closest candidate, is not included in this stability analysis. In terms of stability, methods that adopt the generative model in the counterfactual generation process demonstrate better performance, which suggests the contributions of the generative model to the stability of explanations.

The underlying reasons for these differences can be explained as follows. WACH and DensityGuide-V2 generate counterfactuals by directly optimising randomly initialised parameters. This approach results in a wide range of potential modifications to achieve the desired classifier output, leading to

Table 5.4: Stability evaluation of the explanations provided by the adopted methods across the data sets.

data sets	Stability (the lower the better)					
	WACH	NaiveGuide	LatentGAN	DensityGuide	DensityGuide-V1	DensityGuide-V2
ArticularyWordRecognition	0.09880	\	0.00103	**0.00063**	0.00439	0.32863
BasicMotions	0.08772	\	**0.00069**	0.00076	0.00111	0.26226
CharacterTrajectories	0.29460	\	**0.00068**	0.00093	0.00070	0.48043
Epilepsy	0.31659	\	**0.00069**	0.00119	0.00070	0.71552
ERing	0.02217	\	**0.00045**	0.00084	0.00064	0.20647
Handwriting	0.02640	\	**0.00059**	0.00112	0.00081	0.23712
JapaneseVowels	0.03554	\	**0.00054**	0.00115	0.00089	0.25860
Libras	0.27915	\	0.00058	0.00044	**0.00019**	0.56391
LSST	0.35713	\	0.00071	0.00040	**0.00019**	0.59327
NATOPS	0.07409	\	**0.00124**	0.00139	0.00230	0.32878
PenDigits	0.11143	\	0.00113	0.00058	**0.00020**	0.20039
RacketSports	0.11936	\	0.00153	0.00085	**0.00027**	0.24497
SpokenArabicDigits	0.29442	\	**0.00069**	0.00128	0.00102	0.37400
UWaveGestureLibrary	0.06961	\	0.00033	0.00060	**0.00012**	0.13667
MNIST	0.06888	\	**0.00122**	0.00085	0.00190	0.11403
PdM	0.06251	\	0.00127	**0.00078**	0.00277	0.14840

variability in the final counterfactuals. In contrast, methods that use generative models optimise parameters within the model to transform original random noise into useful counterfactuals that meet the required criteria. This process ensures that the generative model can produce pertinent and actual information regardless of the randomness of the input noise, resulting in more stable and consistent explanations.

As discussed in Chapter 2, the desirable properties of counterfactuals, including the stability mentioned above and diversity, appear to be in conflict. In our proposed framework, counterfactuals are generated by making minimal changes to the original input while simultaneously altering the original prediction label. Under this experimental setting, our approach produces stable and repeatable counterfactuals, ensuring consistency in the final explanations. However, our framework can also generate diverse counterfactuals by simply adjusting the experimental settings. This is achieved by modifying the values of the hyperparameters. For example, by setting the value of λ to a slightly larger number, the generative model can produce counterfactuals that not only meet the requirements of these experimental settings, but also generate various counterfactuals, thereby achieving diversity in explanations.

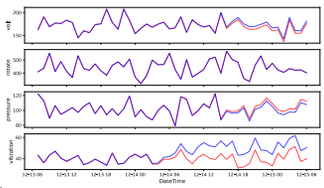
5.6 Using ChatGPT to provide real explanations

Up to this point, we have demonstrated the effectiveness of our approaches in providing more plausible counterfactual explanations compared to existing methods. To understand the working mechanisms and take effective actions, users need to discern and understand the differences between the original inputs and counterfactuals. However, if we can express the explanations in natural language descriptions and tell specifically what needs to do next to the users directly, such kinds of explanation might be more meaningful. Fortunately, recent advances in large language models, such as ChatGPT (OpenAI, 2023), can bridge the gap between the counterfactuals provided and the more meaningful explanations. The advantage of ChatGPT lies in its ability to interpret complex data and produce the outputs in a way that is easily comprehensible to ordinary people. By integrating ChatGPT, we can transform raw data and counterfactuals into clear, actionable insights presented in a user-friendly format, making the explanation process more intuitive and effective for all users.

Fig. 5.7 presents a real example that shows how ChatGPT can be used to provide explanations that are easier to understand. Initially, we input an image into ChatGPT, displaying both the original time series and counterfactuals, along with the necessary textual descriptions to facilitate the interpretation of the image. ChatGPT then analyses the differences between the counterfactual and the original time series, expressing these distinctions in natural language.

Following this, we act as a person who wishes to take action to prevent future failure predicted by the classifier. However, we might lack detailed

Input for ChatGPT:
Image:



Text: The figure presents two distinct of multivariate time series data representing a machine's running state, indicated by measurements of voltage, rotation, pressure, and vibration. These are visualized using different colored lines: the original data is depicted with blue lines, while the counterfactual data is represented by red lines. In the case of the original time series (blue lines), the classifier anticipates a failure of the machine within the next 24 hours. However, the counterfactual time series (red lines) leads the classifier to predict that the machine will continue to operate without failure.

Help to summarise this counterfactual explanation.

ChatGPT:

The primary difference between the original and counterfactual data lies in the vibration levels. The original data showcases sudden spikes and drops in vibration, whereas the counterfactual maintains a more steady and subdued profile. Excessive or erratic vibrations can be an indicator of potential issues, often leading to premature wear and failure of machinery components.

Input for ChatGPT:

I want to avoid the furture failure, do you have some recommended actions?

ChatGPT:

Recommended Actions:

Regular Monitoring: Use vibration sensors and monitoring systems to keep an eye on vibration levels in real-time. Set up alarms or notifications if the machine exceeds safe vibration thresholds.

Balancing: Ensure that all rotating parts of the machine are well-balanced. Imbalances can lead to increased vibration levels.

Foundation & Mounting: Check the machine's foundation and mounting. Ensure it's on a stable surface and tightly secured. If there are any signs of wear, loose bolts, or corrosion, address them immediately.

Maintenance: Schedule routine maintenance and inspections. Over time, wear and tear can lead to parts becoming misaligned or loose, resulting in increased vibrations. Regular maintenance can catch and rectify such issues before they become severe.

Damping: Consider incorporating damping materials or techniques to absorb and dissipate unwanted vibrations

Figure 5.7: Using ChatGPT to provide more comprehensive explanations and suggested actions

knowledge about the machine and therefore be uncertain about the appropriate actions to take. ChatGPT addresses this by providing recommended actions in a manner that is straightforward and easy to understand. It is important to note that these suggestions are quite general in this example, as specific details about the machine were not provided. Providing more comprehensive information, such as a maintenance manual, would allow ChatGPT to offer more targeted and useful action recommendations.

5.7 Summary

In this chapter, two distinct novel approaches were developed to generate counterfactual explanations in the context of MTSC. These approaches aim to provide counterfactuals aligned with the distribution of training data, enhancing their feasibility and practical applicability. Regarding the application of these two methods, the DensityGuide approach is suitable for data sets whose distribution in the latent feature space is effectively characterised by GMMs. In scenarios where GMMs may not adequately capture the distribution of data within the latent features, the LatentGAN approach emerges as a preferable alternative. Experiments conducted with real-world data sets demonstrated that the counterfactuals generated by proposed approaches are more plausible than those provided by existing methods while maintaining comparable levels of similarity and sparsity.

However, the proposed approaches are not without limitations. A significant drawback is their computational expense; generating explanations for each input requires training a separate generative model, leading to substantial time consumption. In the future, we plan to investigate approaches that involve training a universal generative model capable of producing

counterfactual explanations for any instance, given specific desired conditions. Moreover, we note a limitation pertaining to our implicit assumption that training data sets mirror real-world scenarios. In fact, this assumption will not always be realistic. It is important to clarify that while the training data are intended to represent plausible samples that could occur in real-world situations, they are inherently limited. They do not capture the full spectrum of variability and complexity found in real-world conditions. Data sets are often constrained by the scope of their collection and might omit rare or unforeseen scenarios.

Chapter 6

Conclusions and Future Works

The primary objective of this thesis is to enhance the development of XAI within the context of MTSC by providing various types of explanations to meet the diverse needs of stakeholders. The explanations targeted are feature importance and counterfactuals, as these two types of explanations are popular and satisfy the requirements of most individuals. To provide such a range of explanations, this thesis predominantly employs post-hoc explanation techniques. However, a significant challenge encountered with these techniques is the OOD problem, which refers to the fact that the samples used in the explanation process do not align with the distribution of the training data. This OOD issue often leads to unreliable and biased explanations. To mitigate this issue, the main solution of this thesis is the employment of generative models, which are trained to learn the distribution of the training data set and create within-distribution samples for the explanation purpose. This chapter first summarises the main contributions of the works detailed in the previous chapters. Then, it discusses in detail the limitations of current methodologies and outlines potential avenues for future works.

6.1 Main contributions

The main contribution of this thesis is the identification of shortcomings in existing explanation techniques that overlook the OOD issue and the proposal to address this problem through the use of generative models. The detailed contributions are as follows:

6.1.1 A Feature Importance Explanation Framework

In Chapter 3, a feature importance explanation framework is presented that addresses some critical challenges inherent in traditional perturbation-based explanation methods. First, to address the OOD problem, a generative model is designed to approximate the data distribution upon which the classifier is trained. This allows for the creation of perturbed inputs from the training data distribution. Second, as highlighted in Chapter 2.4.4, due to the joint contributions of the input features, obtaining precise importance scores for each feature is an NP-hard problem and intractable, especially for high-dimensional inputs. Therefore, instead of trying to calculate the exact importance scores for each feature as traditional perturbation methods do, the proposed framework aims to search for a set of the most important features that can significantly affect the outputs of classifiers when replaced with plausible alternative values. To imporove the search results and efficiency, this framework is equipped with a greedy-based segmentation and identification search method specifically designed for MTSC problems. Experiments conducted on real-world datasets demonstrate the effectiveness of the proposed framework in identifying the most important features.

6.1.2 SEGAL - A stable LIME framework

LIME is a renowned explanation method that has been widely applied in various applications. However, recent research has highlighted the stability issue with LIME, where the same explanation process produces varying explanation results for the same prediction (Zhou et al., 2021; Visani et al., 2020; Slack et al., 2021). Current solutions attempt to enhance LIME's stability by increasing the number of neighbours (Zhou et al., 2021), employing advanced sampling techniques in the neighbour generation process (Slack et al., 2021), or tuning its hyperparameters (Visani et al., 2020). However, when these solutions are applied to the MTSC problem, their effectiveness is less than satisfactory. The explanations provided remain unstable, indicating that there are other causes for this instability that current solutions overlook. Chapter 4 introduces the SEGAL framework. Unlike traditional neighbour generation methods that simply replace certain features with predefined baselines, which inevitably breaks the temporal dependency of time series data and results in OOD samples, the neighbour generation method within this framework incorporates a generative model, which is trained with the aim of creating within-distribution neighbours. Experiments have demonstrated that the explanations provided with the aim of within-distribution neighbours are much more stable than those of traditional methods. This suggests that the OOD issue is another key reason for unstable explanations. This constitutes the primary contribution of this chapter. Furthermore, the SEGAL framework develops a novel adaptive weighting method to replace the traditional exponential kernel weighting method in the classic LIME framework. The experimental results show that this novel adaptive weighting method improves the efficiency of the hyperparameter tuning process. This represents the second major contribution of this chapter.

6.1.3 Plausible Counterfactual Explanations

Compared to feature importance explanations, counterfactual explanations might be preferred by individuals who lack expert knowledge. However, existing counterfactual methods often fail to account for the plausibility and feasibility of the explanations provided. These methods typically focus on making minimal alterations to the inputs with the aim of altering the classifier's output, ignoring the meaningfulness of explanations or whether the explanations align with the training data distribution. Consequently, the changes to the classifier's outputs are not the result of meaningful or insightful alterations, but rather stem from the OOD issue. As a result, the explanations provided by these approaches can be meaningless and cannot help people make informed decisions. To address this shortcoming of current methods, the main contributions of Chapter 5 involve the development of two innovative counterfactual explanation methods. The core goal of these two methods is to ensure that the counterfactuals provided align with the distribution of the training data, thus increasing their plausibility and applicability. The first method begins with the fitting of a GMM to approximate the density of the training data distribution. Then, it trains a generative model to create counterfactuals in the densest region of the training data manifold. However, for some problems, the data distribution might not be effectively captured by GMMs. Therefore, the second method adopts a GAN architecture. Specifically, a generative model is trained alongside a discriminator to produce counterfactuals indistinguishable from actual time series data, ensuring that these explanations faithfully represent the real data distribution. Experimental evaluations of both methods have demonstrated their effectiveness in providing plausible counterfactual explanations.

6.2 Limitations and Future Works

Although this thesis has contributed valuable insights into XAI within the MTSC domain, it is important to recognise certain inherent limitations in the proposed solutions, which open avenues for future research.

6.2.1 The capacity of time series generative models

A primary contribution of this thesis is the employment of generative models to address the potential OOD problem in the explanation process for MTSC problems. Although experiments on real-world applications showcase the efficacy of proposed methods, such approaches encounter some limitations, primarily concerning the models' capacity. Time series data often exhibit more complex relationships than tabular data, such as temporal dependencies. Designing an effective generative model that can accurately capture the intricacies of complex time series data presents significant challenges. If the generative model cannot capture the complex data distribution of the time series data, the obtained samples might still be OOD, which, in turn, results in unreliable explanations. Furthermore, if the generative model only partially captures the distribution of data points, it may learn only certain patterns within the data set and fail to accurately represent the entire distribution. In such a scenario, although the generated samples are within-distribution, they only reflect a portion of the diversity of the data set. This can result in biased explanations, as the model output may not fully represent the underlying complexities of the data. Consequently, the performance of the framework proposed in Chapter 3 and the SEGAL method in Chapter 4 is inherently upper bounded by the capabilities of the underlying generative models. Therefore, future

efforts could be targeted towards developing more sophisticated time series generative models. Enhancing these models' ability to accurately reflect complex data distributions will be crucial in overcoming current limitations and advancing the reliability and applicability of explanations in the MTSC context.

6.2.2 The computational efficiency of proposed methods

Another limitation of the proposed methods is their time expense. For the frameworks proposed in Chapters 3 and 4, generating within-distribution samples requires running the generative model, which is more time-consuming than traditional methods. To enhance the efficiency of these approaches, future work could explore methods to sample within-distribution samples more efficiently.

Furthermore, the counterfactual explanation methods presented in Chapter 5 also face challenges with respect to time consumption, as they require optimising a generative model for each individual input to be explained. Although this approach is effective, it can be prohibitively time-consuming and may not suit applications that require real-time responses. An innovative solution in the future could involve developing a universal generative model that operates on a 'one-size-fits-all' basis, which is capable of quickly generating counterfactuals for any given input under specific conditions. Implementing such a model could significantly speed up the explanation process, thereby improving its practical utility in scenarios where time efficiency is paramount.

6.2.3 Exploring more types of explanations

Finally, this thesis, which aims to advance XAI within the MTSC context, focuses on feature importance and counterfactual explanations; these may not meet the diverse requirements of all users. The field of XAI is vast and varied, with a spectrum of potential users, from data science professionals requiring technical and detailed interpretations to ordinary people seeking simpler and more intuitive explanations. Therefore, future research should strive to broaden the spectrum of explanatory methods. This could include the development of interactive visualisations, user-friendly interfaces, and context-specific explanations tailored to meet the unique needs of various user groups, thus improving the accessibility and applicability of XAI. One thing that has to be denoted is that regardless of the type of explanation provided, when taking post-hoc techniques, the OOD issue should always be considered. Failing to address the OOD issues can lead to explanations that are biased and potentially misleading.

6.2.4 Qualitative evaluation of the explanations

In this work, we have focused solely on the quantitative evaluation of explanation results from various explanation methods. However, the primary objective of XAI is to provide explanations that are comprehensible to humans. It is therefore essential to assess whether explanations deemed superior by quantitative metrics are truly helpful to users in understanding the model's behaviour. Consequently, future research should develop qualitative evaluation methods to more effectively assess the overall performance of the explanations provided.

Appendices

Appendix A

Glossary of Mathematical Notations

This appendix serves as a glossary of some important mathematical notations used throughout this thesis. It is designed to provide quick reference and facilitate a better understanding of the mathematical expressions and formulas discussed.

Symbols

- $\boldsymbol{x}$
  - **Description:** multivariate time series.
- $f\left(c|\boldsymbol{x}\right)$
  - **Description:** the classifier produces the probability of time series, $\boldsymbol{x}$, belonging to class c.
- $f\left(\boldsymbol{x}\right)$

- **Description:** the classifier produces the label for $\boldsymbol{x}$, which has the highest probability.

- $p(\boldsymbol{x})$
  - **Description:** This represents the probability distribution of $\boldsymbol{x}$.

- $p(\boldsymbol{x}_r|\boldsymbol{x}_{\setminus r})$
  - **Description:** This represents the conditional probability distribution of $\boldsymbol{x}_r$ given $\boldsymbol{x}_{\setminus r}$. It is used to model the dependence of the variable $\boldsymbol{x}_r$ on the other variables represented by $\boldsymbol{x}_{\setminus r}$.

- $G_{\boldsymbol{\theta},\hat{\boldsymbol{\epsilon}}}$
  - **Description:** Generative model designed to learn the training data set distribution, which parameterised by model weights $\boldsymbol{\theta}$ and the dropout rate $\hat{\boldsymbol{\epsilon}}$

Bibliography

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Agarwal, C. and Nguyen, A. (2020). Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. arXiv.

Artelt, A. and Hammer, B. (2020). Convex density constraints for computing plausible counterfactual explanations. In Farkaš, I., Masulli, P., and Wermter, S., editors, *Artificial Neural Networks and Machine Learning – ICANN 2020*, pages 353–365, Cham. Springer International Publishing.

Assaf, R., Giurgiu, I., Bagehorn, F., and Schumann, A. (2019). MTEX-CNN: Multivariate time series explanations for predictions with convolutional neural networks. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 952–957.

Ates, E., Aksar, B., Leung, V. J., and Coskun, A. K. (2021a). Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8. IEEE.

Ates, E., Aksar, B., Leung, V. J., and Coskun, A. K. (2021b). Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., and Keogh, E. (2018). The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*.

Bahri, O., Boubrahimi, S. F., and Hamdi, S. M. (2022). Shapelet-based counterfactual explanations for multivariate time series.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Bento, J., Saleiro, P., Cruz, A. F., Figueiredo, M. A., and Bizarro, P. (2021a). TimeSHAP: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 2565–2573, New York, NY, USA. Association for Computing Machinery.

Bento, J., Saleiro, P., Cruz, A. F., Figueiredo, M. A., and Bizarro, P. (2021b). Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2565–2573.

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pages 359–370.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. (2018). Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Castro, J., Gómez, D., and Tejada, J. (2009). Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Chang, C.-H., Creager, E., Goldenberg, A., and Duvenaud, D. (2019). Explaining image classifiers by counterfactual generation. In *International Conference on Learning Representations*.

Conover, W. J. (1999). *Practical nonparametric statistics*, volume 350. john wiley & sons.

Crabbé, J. and Van Der Schaar, M. (2021). Explaining time series predictions with dynamic masks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2166–2177. PMLR.

Delaney, E., Greene, D., and Keane, M. T. (2021a). Instance-based coun-

terfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pages 32–47. Springer.

Delaney, E., Greene, D., and Keane, M. T. (2021b). Instance-based counterfactual explanations for time series classification. In Sánchez-Ruiz, A. A. and Floyd, M. W., editors, *Case-Based Reasoning Research and Development*, pages 32–47, Cham. Springer International Publishing.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc.

Ding, W., Abdel-Basset, M., Hawash, H., and Ali, A. M. (2022). Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. *Information Sciences*, 615:238–292.

Dong, Y., Sun, Z., and Jia, H. (2006). A cosine similarity-based negative selection algorithm for time series novelty detection. *Mechanical Systems and Signal Processing*, 20(6):1461–1472. Special Issue: Laser Doppler Vibrometry.

Doroudyan, M. H. and Niaki, S. T. A. (2021). Pattern recognition in financial surveillance with the arma-garch time series model using support vector machine. *Expert Systems with Applications*, 182:115334.

Douzal-Chouakria, A. and Amblard, C. (2012). Classification trees for time series. *Pattern Recognition*, 45(3):1076–1091.

Du, Q. and Xu, J. (2021). Model-agnostic local explanations with genetic algorithms for text classification. In *The 33rd International Conference on Software Engineering & Knowledge Engineering*.

Fong, R. C. and Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

Geler, Z., Kurbalija, V., Ivanović, M., and Radovanović, M. (2020). Weighted knn and constrained elastic distances for time-series classification. *Expert Systems with Applications*, 162:113829.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Guidotti, R. (2022). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hamet, P. and Tremblay, J. (2017). Artificial intelligence in medicine.

Metabolism, 69:S36–S40. Insights Into the Future of Medicine: Technologies, Concepts, and Integration.

Hase, P., Xie, H., and Bansal, M. (2021). The Out-of-Distribution problem in explainability and search methods for feature importance explanations. In *Advances in Neural Information Processing Systems*, volume 34.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Höllig, J., Kulbach, C., and Thoma, S. (2022). Tsevo: Evolutionary counterfactual explanations for time series classification. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 29–36.

Ismail, A. A., Gunady, M., Corrada Bravo, H., and Feizi, S. (2020). Benchmarking deep learning interpretability in time series predictions. In *Advances in Neural Information Processing Systems*, volume 33, pages 6441–6452. Curran Associates, Inc.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.

Kamath, U. and Liu, J. (2021). *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*. Springer.

Kampouraki, A., Manis, G., and Nikou, C. (2008). Heartbeat time series classification with support vector machines. *IEEE transactions on information technology in biomedicine*, 13(4):512–518.

Karevan, Z. and Suykens, J. A. (2020). Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9.

Kim, B., Khanna, R., and Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Kim, S., Yi, J., Kim, E., and Yoon, S. (2020). Interpretation of NLP models through input marginalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3154–3167. Association for Computational Linguistics.

Lang, J., Giese, M. A., Ilg, W., and Otte, S. (2023a). Generating sparse counterfactual explanations for multivariate time series. In Iliadis, L., Papaleonidas, A., Angelov, P., and Jayne, C., editors, *Artificial Neural Networks and Machine Learning – ICANN 2023*, pages 180–193, Cham. Springer Nature Switzerland.

Lang, J., Giese, M. A., Ilg, W., and Otte, S. (2023b). Generating sparse counterfactual explanations for multivariate time series. In Iliadis, L., Papaleonidas, A., Angelov, P., and Jayne, C., editors, *Artificial Neural Networks and Machine Learning – ICANN 2023*, pages 180–193, Cham. Springer Nature Switzerland.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., and Detyniecki, M. (2019). The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 2801–2807. AAAI Press.

Le, T.-T.-H., Prihatno, A. T., Oktian, Y. E., Kang, H., and Kim, H. (2023). Exploring local explanation of practical industrial ai applications: A systematic literature review. *Applied Sciences*, 13(9):5809.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, Y.-H., Wei, C.-P., Cheng, T.-H., and Yang, C.-T. (2012). Nearest-neighbor-based approach to time-series classification. *Decision Support Systems*, 53(1):207–217.

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Masini, R. P., Medeiros, M. C., and Mendes, E. F. (2023). Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 37(1):76–111.

Meng, H., Wagner, C., and Triguero, I. (2023). Explaining time series classifiers through meaningful perturbation and optimisation. *Information Sciences*, 645:119334.

Miranda, L. J. V. (2018). PySwarms, a research-toolkit for Particle Swarm Optimization in Python. *Journal of Open Source Software*, 3.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Neuberg, L. G. (2003). Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685.

Nti, I. K., Adekoya, A. F., Weyori, B. A., and Nyarko-Boateng, O. (2022). Applications of artificial intelligence in engineering and manufacturing: a systematic review. *Journal of Intelligent Manufacturing*, 33(6):1581–1601.

OpenAI (2023). ChatGPT (Mar 14 version). https://chat.openai.com/chat. Large language model.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., and Flach, P. (2020). Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 344–350, New York, NY, USA. Association for Computing Machinery.

Qiu, L., Yang, Y., Cao, C. C., Liu, J., Zheng, Y., Ngai, H. H. T., Hsiao, J., and Chen, L. (2021). Resisting Out-Of-Distribution data problem in perturbation of XAI. *arXiv preprint arXiv:2107.14000*.

R. Fernández, R., Martín de Diego, I., M. Moguerza, J., and Herrera, F. (2022). Explanation sets: A general framework for machine learning explainability. *Information Sciences*, 617:464–481.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?" explaining the predictions of any classifier. In *Proceedings of*

the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Roberts, H., Babuta, A., Morley, J., Thomas, C., Taddeo, M., and Floridi, L. (2023). Artificial intelligence regulation in the united kingdom: A path to good governance and global leadership? *Internet Policy Review*, 12(2):1–31.

Rosafalco, L., Manzoni, A., Mariani, S., and Corigliano, A. (2020). Fully convolutional networks for structural health monitoring through multivariate time series classification. *Advanced Modeling and Simulation in Engineering Sciences*, 7(1):38.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017a). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017b). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Serradilla, O., Zugasti, E., Rodriguez, J., and Zurutuza, U. (2022). Deep

learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, 52(10):10934–10964.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In Precup, D. and Teh, Y. W., editors, *The 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*.

Situ, X., Zukerman, I., Paris, C., Maruf, S., and Haffari, G. (2021). Learning to explain: Generating stable explanations fast. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5340–5355, Online. Association for Computational Linguistics.

Slack, D., Hilgard, A., Singh, S., and Lakkaraju, H. (2021). Reliable post hoc explanations: Modeling uncertainty in explainability. In *Advances in Neural Information Processing Systems*, volume 34, pages 9391–9404. Curran Associates, Inc.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Sturmfels, P., Lundberg, S., and Lee, S.-I. (2020). Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In Precup, D. and Teh, Y. W., editors, *The 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Tadapaneni, N. R. (2019). Artificial intelligence in finance and investments. *International Journal of Innovative Research in Science, Engineering and Technology*, 9(5).

Vafa, K., Deng, Y., Blei, D., and Rush, A. (2021). Rationales for sequential predictions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10314–10332, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

van der Waa, J., Nieuwburg, E., Cremers, A., and Neerincx, M. (2021). Evaluating xai: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, 291:103404.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Visani, G., Bagli, E., and Chesani, F. (2020). Optilime: Optimized lime explanations for diagnostic computer algorithms. *Proceedings of the CIKM 2020 Workshops*, abs/2006.05714.

Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and

the gdpr. *Harvard Journal of Law & Technology (Harvard JOLT)*, 31(2):841–888.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, Z., Hooker, G., and Wang, F. (2021). S-lime: Stabilized-lime for model explanation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 2429–2438, New York, NY, USA. Association for Computing Machinery.