Università degli Studi di Trieste

Ph.D. in Applied Data Science & Artificial Intelligence

XXXVIII cycle

Interpretable Time Series Classification via Temporal Logic Embeddings

Candidate
Irene Ferfoglia

Supervisors
Prof. Luca Bortolussi
Prof.ssa Laura Nenzi

Summary

Time series classification has become increasingly dominated by deep learning models that achieve high predictive performance but offer limited transparency. In safety-critical and regulated domains, such as healthcare, industrial monitoring, and cyber-physical systems, the lack of interpretability undermines trust, accountability, and practical adoption. Existing explainability approaches are predominantly post-hoc: they attempt to justify decisions after training, often producing explanations that are unstable or weakly connected to the model's actual reasoning process.

This thesis addresses the challenge of designing a classification framework in which interpretability is not added after training, but embedded directly into the model itself. The central idea is to express predictions in terms of human-understandable temporal concepts. To this end, the thesis leverages Signal Temporal Logic (STL), a formal language capable of describing time-dependent behaviours such as persistence, bounded responses, and structured temporal patterns that naturally arise in real-world signals.

The main outcome of this work is STELLE (Signal Temporal Logic Embedding for Logically-grounded Learning and Explanation), a neuro-symbolic framework that connects continuous time series data with symbolic temporal concepts. Instead of relying on external explanation methods, STELLE produces predictions and explanations through the same internal mechanism. It supports both local explanations for individual instances and global explanations that characterise class-level temporal patterns, ensuring that explanations remain faithful to the classifier's decisions.

Extensive empirical evaluation on univariate and multivariate benchmarks shows that STELLE achieves competitive predictive performance while providing concise and semantically meaningful explanations. The analysis highlights the trade-offs between expressiveness, interpretability, and accuracy, and demonstrates that logically grounded explanations can be integrated into modern learning architectures without sacrificing effectiveness. Overall, this thesis contributes a principled approach to interpretable and trustworthy time series learning.

Contents

Chapter 1

Introduction

Time series data is fundamental to a wide range of safety-critical domains, including medical diagnostics, autonomous and cyber-physical systems, industrial monitoring, and environmental control. In these settings, decisions informed by data-driven models can have direct consequences for human safety, regulatory compliance, and economic stability. While modern deep learning architectures achieve state-of-the-art performance in classifying and forecasting time series data, their black-box nature poses a substantial barrier to real-world deployment. The opacity of such models limits transparency and accountability, raises ethical and legal concerns under emerging regulatory frameworks, undermines user trust, and prevents domain experts from validating, contesting, or learning from model predictions.

The challenge of explainability is particularly acute for time series data. Unlike images or text, where salient regions or tokens often have an intuitive semantic interpretation, temporal patterns are inherently abstract and context-dependent. Relevant information may be encoded in trends, oscillations, delays, persistence, or ordered sequences of events that are not directly observable from individual time points. Interpreting these patterns typically requires substantial domain expertise, making it difficult for generic explanation methods to provide meaningful insight. As a result, many existing Explainable AI (XAI) techniques for time series focus on post-hoc explanations, such as saliency maps or attribution scores highlighting influential time points or subsequences after a model has been trained. Although useful for exploratory analysis, these approaches have been criticised for their instability, limited semantic grounding, and, most importantly, their lack of guaranteed faithfulness to the model's true decision process.

These limitations have motivated increasing interest in models that are interpretable by design, where transparency is embedded directly into the learning architecture rather than imposed after the fact. In such models, predictions are produced through intermediate representations that correspond to human-understandable concepts, enabling explanations that are inherently aligned with the model's internal reasoning. However, designing interpretable architectures for time series remains particularly challenging, as temporal concepts are rarely local or static. Instead, they are defined by structured relationships over time, such as ordering, duration, recurrence, and temporal constraints, which are difficult to capture using standard neural representations alone.

At the same time, purely symbolic approaches to time series analysis, while offering transparency and formal guarantees, often struggle to scale to noisy, high-dimensional,

and heterogeneous data. Hand-crafted rules or logical specifications may fail to capture the variability and complexity of real-world signals, limiting their predictive performance and adaptability. This creates a persistent tension between statistical expressiveness and symbolic interpretability. For time series data, this tension is especially pronounced: meaningful patterns are not defined solely by local signal values, but by higher-level temporal relationships that unfold across multiple time scales.

Bridging this gap requires models that can learn directly from raw temporal data while reasoning in terms of structured, human-understandable temporal concepts. Such models must combine the representational power and scalability of deep learning with the semantic clarity, compositionality, and verifiability of symbolic reasoning. Addressing this challenge is essential for developing time series models that are not only accurate, but also trustworthy, auditable, and suitable for deployment in safety-critical environments.

1.1 Research Objective

The central objective of this thesis is to develop a time series classification framework that reconciles high predictive performance with native interpretability. Rather than treating explainability as an external diagnostic tool applied after training, the goal is to design models in which explanations are an integral part of the decision-making process. In particular, model predictions should be explained directly in terms of semantically grounded temporal concepts that are meaningful to human users, enabling transparent reasoning, verification, and insight extraction.

To this end, the thesis investigates how formal temporal logic can be systematically integrated into modern learning architectures for time series. Temporal logic provides a structured and expressive language for describing temporal patterns such as persistence, ordering, recurrence, and bounded temporal constraints, which are central to many real-world time series phenomena. By embedding time series data into a symbolic concept space defined by temporal logic specifications, the proposed framework aims to bridge the gap between continuous neural representations and discrete, human-interpretable temporal abstractions.

A key objective is to enable the joint optimisation of predictive accuracy and interpretability. This involves learning both how temporal concepts relate to raw time series data and how these concepts contribute to the final classification decision. As a result, explanations are not derived through post-hoc approximation methods, but emerge directly from the model's internal structure and learned parameters. This joint optimisation ensures that explanations are faithful to the classifier's actual reasoning process, while remaining stable and semantically meaningful.

More broadly, this thesis seeks to contribute to the development of neuro-symbolic approaches for time series analysis that combine the scalability and expressive power of deep learning with the transparency, compositionality, and formal grounding of symbolic reasoning. By focusing on time series classification, the work aims to demonstrate that it is possible to achieve competitive performance on complex temporal data while simultaneously providing explanations that are suitable for deployment in safety-critical and high-stakes application domains.

1.2 Contributions

This thesis makes the following main contributions:

- **The STELLE Framework and STL - Based Trajectory Embedding**. We introduce STELLE (Signal Temporal Logic Embedding for Logically-grounded Learning and Explanation), an interpretable-by-design neuro-symbolic architecture for time series classification. At its core, STELLE relies on a novel trajectory embedding kernel based on STL quantitative robustness, which maps raw time series into a semantic concept space defined by temporal logic formulae, thereby tightly coupling prediction and explanation within a single learning framework.
- **Integrated Local and Global Explanations**. We develop an explanation mechanism that produces both instance-level and class-level explanations from the same model, without requiring additional post-hoc procedures.
- **Extensive Empirical Evaluation**. We conduct a comprehensive experimental study across univariate and multivariate benchmarks, including ablation and sensitivity analyses, demonstrating that STELLE achieves competitive accuracy while producing concise and faithful symbolic explanations.

Publications

The body of work developed during this doctoral research is consolidated in the STELLE framework, which unifies several years of research on time series explainability, temporal logic, and concept-based reasoning. The journal submission listed below represents the integrated and mature form of these contributions.

- [1] Irene Ferfoglia, Simone Silvetti, Gaia Saveri, Laura Nenzi, and Luca Bortolussi. Guided by stars: Interpretable concept learning over time series via temporal logic semantics, 2025.

A preliminary version of this research was published as:

- [2] Irene Ferfoglia, Gaia Saveri, Laura Nenzi, and Luca Bortolussi. Ecats: Explainable-by-design concept-based anomaly detection for time series. In Tarek R. Besold, Artur d'Avila Garcez, Ernesto Jimenez-Ruiz, Roberto Confalonieri, Pranava Madhyastha, and Benedikt Wagner, editors, *Neural-Symbolic Learning and Reasoning*, pages 175–191, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-71170-1.

Several components presented in this thesis, including the expanded theoretical analysis, extensive experimental evaluation, and the ablation and sensitivity studies, are original to this work and were not included in the published or submitted manuscripts.

Roadmap

This thesis is organised into two main parts.

Part I (Background and State of the Art) provides the theoretical and methodological foundations necessary to contextualise the proposed research. Chapter 2 reviews the evolution of machine learning, focusing on the rise of deep learning and the associated challenges related to model complexity and interpretability. Chapter 3 introduces time series analysis, covering fundamental definitions, classical approaches, and the state of the art in time series classification. Chapter 4 examines the interpretability problem in modern machine learning, discussing Explainable Artificial Intelligence, regulatory perspectives, and existing explanation paradigms, with particular attention to time series data. Chapter 5 introduces temporal logic, progressing from symbolic logic to Signal Temporal Logic (STL) and its quantitative semantics. Chapter 6 discusses neuro-symbolic artificial intelligence and concept-based models as a framework for integrating statistical learning with symbolic reasoning. Part I concludes with Chapter 7, which synthesises the reviewed literature and identifies the research gaps addressed in this thesis.

Part II (Core Contribution) presents the main contributions of this thesis and is organised into two complementary components: the proposed method and its empirical evaluation. Chapter II introduces the STELLE framework in its entirety, formalising the problem of logically grounded interpretable learning and presenting Signal Temporal Logic (STL) as the underlying concept language. It details the construction and selection of interpretable concept sets, introduces the trajectory–formula embedding kernel for jointly representing STL formulae and time series trajectories, and describes the STELLE architecture, including its embedding layers, discriminability mechanism, concept relevance modelling, and interpretability-by-design properties. The chapter further presents the explanation framework integrated into the model, covering the extraction of local and global explanations and the post-processing steps adopted to improve their readability and semantic clarity.
Chapter 9 reports the experimental evaluation of STELLE, including the datasets considered, training protocols, baseline methods, and quantitative results on both classification performance and explanatory quality. Chapter 10 presents ablation and sensitivity studies, analysing the impact of key design choices related to concept set construction, kernel formulation, architectural components, and explanation mechanisms.

Finally, Chapter 11 concludes the thesis by summarising the main contributions, discussing limitations, and outlining directions for future research.

Part I

Background and State of the Art

Chapter 2

The evolution of Machine Learning

The development of Artificial Intelligence (AI) has long been driven by the goal of building machines capable of learning from data. Over several decades, this effort has led not only to new algorithms, but also to a shift in how intelligence itself is understood, going from systems governed by hand-crafted rules to models that infer patterns, make predictions, and adapt based on experience. Machine Learning (ML) lies at the centre of this shift, providing the theoretical foundations and practical methods that enabled AI to move towards data-driven generalisation [3, 4, 5].
This chapter reviews the evolution of machine learning, from early symbolic approaches and statistical methods to the rise of deep learning, which now plays a dominant role in the field [6]. We outline the core principles shared by learning systems, highlight the main historical milestones, and discuss the specialised architectures developed to address increasingly complex data types. Placing these developments in context helps clarify both the strengths of modern machine learning and the open challenges it faces, in particular the tension between predictive performance and interpretability that motivates the research presented in this thesis [7].

2.1 Core Principles

At its core, machine learning addresses the problem of *generalisation*, that is, learning from a finite set of examples in a way that enables accurate predictions on previously unseen data [8]. This goal is common to all machine learning approaches and distinguishes learning systems from those that simply store or recall information. Successful generalisation requires a model to capture the underlying structure of the problem while remaining robust to the noise that is inevitably present in real-world data [9].

This behaviour is mainly governed by the *bias–variance trade-off*, which characterises a model's tendency to underfit or overfit the training data [10]. Models that are too simple fail to represent important patterns, resulting in underfitting and poor performance on both training and test data. At the opposite extreme, overly flexible models may fit the training data too closely, including noise, and consequently generalise poorly. Designing effective learning algorithms therefore amounts to balancing model expressiveness and robustness, so that the true signal is captured without overreacting to random fluctuations [9, 11].

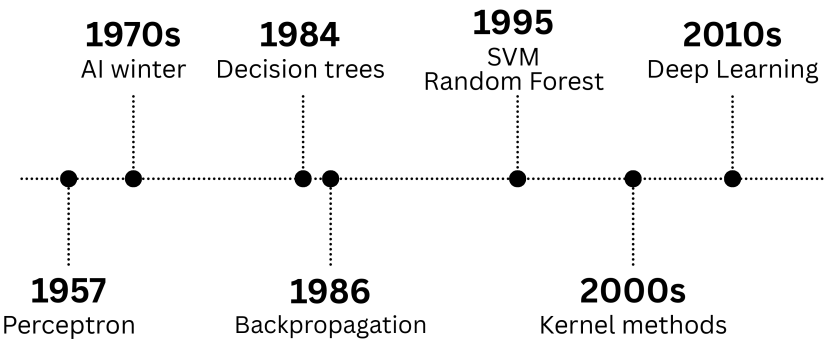


Figure 2.1: A schematic timeline of major developments in machine learning, highlighting key algorithmic milestones and shifts in dominant paradigms.

Different learning paradigms approach this challenge in distinct ways. In *supervised learning*, models are trained using labelled data to learn a mapping from inputs to desired outputs. *Unsupervised learning* removes this guidance, instead aiming to uncover latent structure, regularities, or low-dimensional representations in the data. *Reinforcement learning* addresses learning in a sequential setting, where an agent interacts with an environment and improves its behaviour based on reward signals [12]. Despite their differences, all three paradigms are united by the same objective, extracting knowledge from experience that generalises beyond the observed data.

2.2 Historical Milestones

The history of machine learning is characterised by alternating periods of rapid progress and enthusiasm, followed by phases of scepticism and reduced funding, commonly referred to as "AI winters" [13]. These cycles were largely driven by the gap between ambitious expectations and the practical limitations of available models, data, and computational resources.

Figure 2.1 presents a high-level overview of key developments in machine learning, illustrating major paradigm shifts and the emergence of influential methods over time.

The early period of the *1950s and 1960s* saw the introduction of the *Perceptron* by Frank Rosenblatt [14], a single-layer neural network capable of learning simple binary classification tasks. Beyond its immediate algorithmic contribution, the Perceptron was important as the first concrete instantiation of the *connectionist paradigm*, demonstrating that learning could emerge from networks of simple computational units. This initial optimism was, however, tempered by its inherent limitations. The influential analysis of Minsky and Papert [15] showed that single-layer networks were unable to represent basic non-linear functions, such as the XOR relation. This critique exposed a fundamental architectural shortcoming and led to a decline in neural network research, with attention shifting back towards symbolic, rule-based approaches that dominated AI for much of the following decade.

The *1980s* marked a revival of connectionist methods, driven largely by the rediscovery and popularisation of the backpropagation algorithm [16]. Backpropagation provided an efficient mechanism for training *Multi-Layer Perceptrons* (MLPs), enabling networks to overcome the expressiveness limits of single-layer models and to learn more

complex, hierarchical representations. During the same period, alternative learning paradigms gained traction, including *decision trees*, valued for their interpretability and ability to model non-linear decision boundaries through simple rules, and instance-based methods such as *k-nearest neighbours*, which offered a flexible and intuitive non-parametric approach to classification.

The *1990s and 2000s* were dominated by *kernel methods*, most notably *Support Vector Machines* (SVMs) [17]. Their success was largely due to the *kernel trick*, which made it possible to handle highly non-linear problems by implicitly mapping data into high-dimensional feature spaces without incurring prohibitive computational costs. In parallel, ensemble methods such as Random Forests [18] became increasingly popular. By aggregating many high-variance decision trees, typically through bagging and random feature selection, these models achieved strong performance and robustness via a form of collective averaging.

The most recent phase, beginning in the *2010s*, has been shaped by the rapid rise of *deep learning*. This shift was enabled by several converging factors, including the availability of large-scale labelled datasets (such as ImageNet [19]), advances in parallel computing through GPUs, and architectural innovations like Rectified Linear Units (ReLU) and Dropout that improved the stability of training deep networks. Deep neural models achieved dramatic performance gains in a range of previously challenging tasks. Notable examples include the success of *Convolutional Neural Networks* (CNNs) in image recognition [20], where spatial hierarchies are learned directly from raw pixels, and the application of *Recurrent Neural Networks* (RNNs) to sequential domains such as speech recognition and machine translation.

2.3 The Deep Learning Revolution

While the theoretical foundations for deep learning, i.e., artificial neural networks and the backpropagation algorithm, were established in the previous century, their convergence with three critical enablers in the early 2010s catalysed the modern revolution: massive datasets, specialised hardware, and software ecosystems.

Deep Learning is built upon the foundational concept of Artificial Neural Networks (ANNs). The field's simplest unit, the Perceptron, performs a weighted sum of its inputs and passes the result through a step function to produce a binary output. While powerful for linearly separable problems, its limitations were stark. The advent of the multi-layer perceptron, comprising an input layer, one or more hidden layers, and an output layer, overcame this by enabling the learning of non-linear decision boundaries. The critical algorithm that made training these networks feasible is *backpropagation*, which efficiently calculates the gradient of the loss function with respect to each weight by applying the chain rule, propagating error signals backwards through the network to guide weight updates.

The training of deep networks is computationally intensive, relying on massive matrix multiplications. The adoption of *Graphics Processing Units* (GPUs), with their thousands of cores designed for parallel processing, provided a million-fold increase in computational speed over CPUs, making the training of deep models practically feasible. This hardware revolution was accompanied by the rise of open-source software frameworks like TensorFlow [21] and PyTorch [22], which abstracted away the low-level

complexities of gradient computation and provided robust, scalable environments for research and development, democratising access to state-of-the-art tools.

A key component enabling non-linearity in these networks is the *activation function*. Functions such as the logistic *sigmoid* and the *hyperbolic tangent* (tanh) were historically common, but they are susceptible to the *vanishing gradient* problem in deep architectures. This issue arises when gradients become progressively smaller as they are propagated backwards through many layers, eventually approaching zero and preventing the earlier layers from learning effectively. The *Rectified Linear Unit* (ReLU) [23], which outputs the input directly if positive and zero otherwise, largely mitigated this issue and became the default activation in many architectures due to its computational simplicity and improved gradient flow.

Training deep neural networks also requires careful choices in both output activations and optimisation strategies. In classification tasks, the *softmax* function is typically employed in the final layer to transform raw scores into a normalised probability distribution. Beneath this, learning is driven by variants of gradient descent. Although standard stochastic gradient descent provides the conceptual foundation, modern deep learning relies on adaptive methods such as *Adam* [24], which adjust learning rates dynamically for each parameter to improve stability and convergence. Given the expressive capacity of deep models, mitigating overfitting and stabilising training are equally important. *Dropout* [25] introduces stochasticity by randomly deactivating a proportion of neurons during training, encouraging the network to rely on distributed, robust representations. Complementing this, *batch normalisation* [26] normalises intermediate activations, smoothing the loss landscape and accelerating training by reducing internal covariate shift. Together, these optimisation and regularisation techniques underpin the practical success of contemporary deep learning.

The field has progressed through the development of specialised neural network architectures designed for specific data modalities. The release of large-scale, labelled datasets like ImageNet [19] was instrumental in this progress, providing the fuel for complex models. For processing grid-like data such as images, convolutional neural networks were revolutionary. Pioneered by LeCun et al. [27] for digit recognition, their core operations (convolution and pooling) exploit translational invariance, allowing them to hierarchically learn spatial features, from edges to complex objects. This culminated in deep CNNs [20] that achieved outstanding performance on image classification tasks. For sequential data like text or time series, recurrent neural networks such as the Elman network [28], were designed with loops to persist information across time steps. However, simple RNNs suffer from the vanishing gradient problem when learning long-range dependencies. This was addressed by more complex gated units, most notably the *Long Short-Term Memory* (LSTM) network [29] and the *Gated Recurrent Unit* (GRU) [30], which use gating mechanisms to selectively remember or forget information over long periods. Such gates selectively retain or erase information, ensuring gradients can propagate over long time spans.

A paradigm shift occurred with the introduction of the *attention mechanism* [31], which allowed models to dynamically focus on the most relevant parts of the input sequence when producing an output. This concept was perfected in the *Transformer* architecture [32], which relies solely on self-attention mechanisms, dispensing with recurrence entirely. This design enables significantly greater parallelisation during

training and has become the dominant architecture not only for natural language processing but also for vision and multimodal tasks. Underpinning the success of these architectures is the concept of *representation learning*, where models automatically discover the hierarchical representations needed for feature detection or classification from raw data. This is exemplified by *embeddings* [33] (dense, lower-dimensional vector representations of discrete entities), *autoencoders* [34] (networks trained to reconstruct their input through a compressed bottleneck layer), and more recently, *contrastive learning* [35] (which learns representations by contrasting positive and negative data pairs).

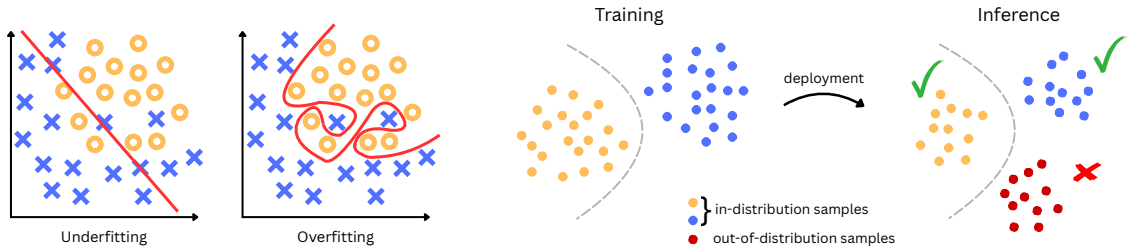
2.4 Fundamental challenges

The remarkable progress in machine learning, particularly in deep learning, has unlocked unprecedented capabilities. However, this progress has also exposed and amplified a set of fundamental challenges that continue to limit the applicability, reliability, and trustworthiness of ML systems [5, 11]. These challenges are especially pronounced in critical domains where decisions have significant consequences.

The interpretability crisis. The most widely discussed challenge stemming from the deep learning revolution is the interpretability crisis [7, 36, 37]. As models have grown more complex, from intelligible linear models and decision trees to deep neural networks with millions or even billions of parameters, their internal workings have become increasingly opaque, offering little visibility into how decisions are formed [38]. This opacity creates a critical barrier to adoption in fields like healthcare, finance, and criminal justice, where understanding the why behind a decision is as important as the decision itself [39]. It hinders debugging, complicates the detection of biased reasoning, and undermines user trust, creating a central tension between model performance and model transparency.

Data dependency and quality. The performance of modern ML models is heavily dependent on the data they are trained on, giving rise to several related challenges [40, 41, 42, 43]:

- *Data hunger.* Deep learning models typically require vast amounts of labelled training data to achieve high performance, which can be expensive, time-consuming, or simply impossible to acquire in certain domains (e.g., rare diseases).
- *Data bias.* Models can easily learn and amplify spurious correlations and societal biases present in the training data. This leads to issues of fairness and discrimination, where models perform poorly for under-represented subgroups.
- *The Garbage In, Garbage Out principle.* Noisy, incorrectly labelled, or unrepresentative data will inevitably lead to an unreliable and untrustworthy model, regardless of the sophistication of the algorithm.



(a) Underfitting vs. overfitting.

(b) Illustration of OOD generalisation.

Figure 2.2: Examples of common generalisation failures in machine learning.

Computational and environmental costs. The pursuit of state-of-the-art performance has led to an exponential growth in model size and computational requirements. Training large models, such as modern large language models, requires immense computational resources, leading to high financial costs and a significant environmental footprint due to the associated energy consumption. This creates barriers to entry for smaller research institutions and raises ethical questions about the sustainability of the field's current trajectory [44, 45, 46].

Generalisation and robustness. A core aim of ML is to build models that generalise well to unseen data. However, several issues complicate this [11, 47]:

- *Overfitting.* Complex models can memorise noise and specific examples in the training data rather than learning the underlying pattern, causing poor performance on new data (Figure 2.2a).
- *Adversarial attacks.* Models are often surprisingly fragile. Small, carefully crafted perturbations to input data, imperceptible to a human, can cause the model to make highly confident incorrect predictions, revealing a lack of robustness and posing security risks [48].
- *Out-of-distribution generalisation (OOD).* Models frequently fail when faced with data that comes from a different distribution than the training data (e.g., a model trained on daytime images failing on nighttime images), highlighting a lack of true reasoning and understanding [49] (Figure 2.2b).

The Challenge of Temporal Data

While machine learning and deep learning models have solved innumerable complex problems, applying them to time series data presents a unique and formidable set of challenges compared to working with static, independent data points [50]. The core difficulty lies in the violation of the fundamental assumption of independence and identical distribution (i.i.d.) that underpins many classical ML algorithms. In time series, observations are inherently temporally dependent: the value at time t is almost always influenced by values at times $t-1, t-2$, etc. This temporal structure contains critical information (e.g., trends, seasonality, autocorrelation) that must be explicitly modelled. Furthermore, time series can be plagued with non-stationarity,

where their statistical properties (like mean and variance) change over time. Noise, missing values, and the need to model variable-length sequences add further complexity. Consequently, specialised architectures like LSTMs [29], with their gating mechanisms to control information flow, and Temporal Convolutional Networks (TCNs) [51], which use dilated causal convolutions, have been developed to directly capture these long-range, hierarchical temporal dependencies [52, 53]. However, these deep learning solutions often achieve performance at the expense of interpretability, making time series learning a distinct and critically important area of research where the tension between accuracy and transparency is acutely felt.

Concluding Remarks

The historical development of machine learning shows a steady increase in model expressiveness, moving from rule-based systems to deep architectures capable of capturing highly complex patterns. However, this increase in capacity has also made models harder to analyse and validate. Questions of generalisation, robustness, and interpretability have not disappeared; if anything, they have become more difficult to address as models grow in complexity.

These issues become even more pronounced in sequential settings. Temporal dependence, non-stationarity, and long-range interactions introduce challenges that cannot be fully understood through general machine learning principles alone. For this reason, the next chapter turns specifically to time series classification, examining the distinctive characteristics of temporal data and clarifying why interpretability in this domain requires dedicated methodological attention.

Chapter 3

Time Series Analysis

Time series data pervades modern science and industry, from financial markets and climate systems to medical diagnostics and industrial sensors. Unlike static datasets, time series encode temporal dynamics such as trends, seasonality, dependencies, and non-stationarities, that require specialised analytical frameworks. This chapter provides a comprehensive overview of time series analysis, beginning with foundational concepts and classical statistical methods before surveying the state of the art in machine learning approaches. We examine the unique challenges posed by temporal data, and we establish the technical landscape within which the interpretability challenges addressed by this thesis become particularly acute: as time series models have grown more powerful, they have simultaneously become more opaque, creating an urgent need for transparent, semantically grounded approaches to temporal reasoning.

3.1 Definition and properties

Time series data arises whenever observations are collected sequentially over time, capturing the dynamic evolution of a process or system. This form of data is pervasive across scientific and industrial domains, from finance and medicine to engineering and environmental monitoring. Its defining characteristic is the inherent temporal ordering of observations: the sequence is not arbitrary, and the relative position of each data point carries essential information about the underlying dynamics.

As a consequence, time series typically exhibit *temporal dependence*, whereby the value observed at time t is statistically related to past observations (e.g., at $t-1, t-2, \ldots$). Beyond this dependence structure, real-world time series often display additional properties such as *trend*, corresponding to long-term increases or decreases in the signal, and *seasonality*, characterised by regular and repeating patterns driven by temporal cycles such as hours, days, or seasons. Moreover, many time series are *non-stationary*, meaning that their underlying statistical properties—such as mean and variance—evolve over time, posing significant challenges for both modelling and inference [54, 55].

Time series can be categorised along several dimensions. A primary distinction is between *univariate* time series, which consist of a single observed value per time stamp, and *multivariate* time series, which contain multiple, potentially interdependent variables measured simultaneously [56, 57]. They can also be classified by their sampling: *regularly sampled* series have equidistant time intervals (e.g., hourly stock

prices, daily temperature), whereas *irregularly sampled* series have non-uniform time gaps between observations (e.g., medical lab tests, network event logs). Finally, the values themselves can be *continuous* (real-valued measurements like sensor data) or *symbolic* (discrete symbols from a finite alphabet, often derived from preprocessing continuous data) [58].

Modelling time series presents multiple distinct challenges. *Missing data* is a frequent issue, whether due to sensor failure or irregular sampling, and must be handled via imputation or model-based approaches. The presence of *noise* (random variations or errors that obscure the underlying signal) requires robust modelling and filtering techniques. *Scaling* multivariate series where variables have different units and value ranges is often necessary. Furthermore, the *high dimensionality* of long and/or multivariate series can lead to computational bottlenecks and the *curse of dimensionality*, that is, the phenomenon where data becomes increasingly sparse as the number of dimensions grows, making it difficult to find meaningful patterns and requiring exponentially more data for reliable statistical inference, therefore necessitating efficient algorithms and feature reduction [59, 60].

The applications of time series modelling are vast and often safety- or mission-critical. In finance, time series analysis underpins tasks such as stock price forecasting, market risk assessment, and algorithmic trading. In healthcare, it plays a central role in electrocardiogram (ECG) analysis, sleep stage scoring, patient monitoring, and epidemic forecasting. Industrial applications include predictive maintenance through sensor-based fault detection and supply chain optimisation via demand forecasting. A prominent modern example is Human Activity Recognition [61], where multivariate time series collected from wearable or smartphone sensors are classified into activities such as walking, running, or sitting.
These diverse application domains place stringent requirements on time series models, demanding not only accuracy and scalability, but also robustness, interpretability, and the ability to capture complex temporal dependencies.

3.2 Classical methods

Classical statistical methods provide a strong foundation for time series analysis. *Autoregressive* (AR) models predict future values based on a linear combination of past values. This concept was extended into the *Autoregressive Integrated Moving Average* (ARIMA) framework [55], which incorporates differencing to handle non-stationarity and moving averages of past forecast errors. For multivariate series, *Vector Autoregression* (VAR) models capture the linear interdependencies between multiple time series [62]. *Hidden Markov Models* (HMMs) [63] are a generative approach that assumes the observed series is produced by a underlying, unobserved Markov process with a finite number of states, making them highly effective for tasks like speech recognition. *Kalman filters* [64] are another pioneering technique, providing an efficient recursive algorithm to estimate the state of a dynamic system from a series of noisy measurements, with extensive applications in tracking and control systems.

The limitations of these classical methods, particularly their often-linear assumptions and difficulty capturing complex non-linear patterns, prompted a transition to machine learning approaches. Early ML methods for time series included *distance-based*

approaches for classification and clustering, using specialised metrics like Dynamic Time Warping (DTW) [65] to compare series of different lengths and speeds. The concept of *shapelets* (discriminative, phase-independent subsequences) was developed for interpretable time series classification [66]. A common pipeline involved manual *feature extraction* (e.g., deriving statistical properties like variance, Fourier coefficients), which would then be fed into standard classifiers like SVMs or Random Forests.

While these classical and early machine learning approaches provide a strong methodological foundation, the increasing complexity, scale, and non-linearity of real-world time series data has driven the field towards more sophisticated modelling paradigms, as surveyed in the following state of the art.

3.3 State of the Art

As discussed, Time Series Classification (TSC) is a central problem in machine learning, with wide-ranging applications in finance, healthcare, industrial monitoring, and human activity recognition. The proliferation of sensors and IoT devices has led to an explosion of such data, making scalable and interpretable analysis more critical than ever. Unlike static feature vectors, time series consist of ordered, and often multivariate, observations that encode both temporal dependencies and dynamic patterns. This sequential nature introduces unique challenges such as variable length, temporal distortions, non-stationarity, and high dimensionality.

Distance-Based Approaches. One of the earliest and most widely used approaches to TSC is based on distance measures. The k-*Nearest Neighbour* (k-NN) classifier with *Dynamic Time Warping* (DTW) remains a remarkably strong baseline [67]. DTW aligns sequences by allowing non-linear warping in the time dimension, enabling robust classification in the presence of local misalignments, though its quadratic complexity $O(n^2)$ limits scalability for long series. Numerous variations exist, such as constrained DTW, weighted DTW, and derivative DTW, each addressing different aspects of similarity. Other distance measures include *Euclidean distance* (fast but alignment-sensitive) and *Edit Distance with Real Penalty* (ERP) [68], which generalises edit distance for real-valued signals [67]. While distance-based approaches are simple and effective, their scalability is limited by the need to compute pairwise distances, which becomes prohibitive for large datasets. Ensemble methods built on these measures, such as Proximity Forest [69], have been developed to improve accuracy and robustness while mitigating the inherent limitations of single distance measures.

Feature-Based Approaches. To address scalability, *feature-based methods* transform raw time series into discriminative representations that can be used with classical machine learning algorithms. Feature engineering has included statistical summaries (mean, variance, autocorrelation), frequency-domain features (Fourier or wavelet transforms), and symbolic approximations such as *Symbolic Aggregate approXimation* (SAX) and *Symbolic Fourier Approximation* (SFA) [70, 71]. More advanced methods include *shapelets* [66, 72], discriminative subsequences that capture local patterns strongly correlated with class labels. Shapelets provide intuitive interpretability by explicitly linking subsequences to class membership, though discovering optimal

shapelets can be computationally expensive.
Ensemble-based feature methods, such as the *Hierarchical Vote Collective of Transformation-based Ensembles* (HIVE-COTE) and its successor HIVE-COTE 2.0 [73], represent a significant advance by combining multiple heterogeneous representations (including interval-based, shapelet-based, and spectral features) into a meta-ensemble, achieving state-of-the-art accuracy for several years. Other notable methods include the *Random Interval Spectral Ensemble* (RISE) [74] and the broadly adopted Random Forest algorithm applied to extracted features.

Model-Based Approaches. Model-based methods learn generative or probabilistic models tailored to time series. *Hidden Markov Models* (HMMs) have historically been used for speech and biological sequence classification by capturing latent state transitions. Autoregressive models (AR, ARIMA, VAR) and Gaussian processes have also been applied for both forecasting and classification [75]. These methods excel when the generative assumptions match the data but often struggle with complex or noisy real-world signals.

Deep Learning Approaches. The availability of large benchmark datasets (e.g., UCR/UEA archives [76]) has catalysed the rise of *deep learning* for TSC, shifting the paradigm from hand-crafted features to learned representations. Convolutional Neural Networks (CNNs) can capture local temporal motifs, while Residual Networks (ResNets) and InceptionTime architectures [50] provide hierarchical feature extraction. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks model sequential dependencies but are often outperformed by CNNs in practice. More recently, self-attention mechanisms and Transformers have shown promise for capturing long-range temporal dependencies [77, 78] though their quadratic complexity has led to efficient adaptations like Informer [79] and PatchTST [80] for longer sequences. Other notable architectures include TapNet [81], which incorporates attentional prototypes, and the highly accurate convolution-based ensemble MultiRocket [82]. Despite their accuracy, deep models face significant drawbacks: they often require large datasets, are computationally expensive, and often lack interpretability, which limits their adoption in domains where transparency is critical.

Embedding-Based and Representation Learning Methods. A recent trend involves *embedding-based methods*, which map time series into fixed-length vector representations suitable for standard classifiers. Techniques include autoencoder-based embeddings, random convolutional kernels, and contrastive self-supervised learning frameworks [83, 84]. These approaches enable scalable learning by decoupling representation learning from classification, and they can generalise across domains with transfer learning. Embedding approaches are particularly attractive for large, heterogeneous datasets, though they inherit the black-box nature of the underlying representation [85] as the learned embeddings, while powerful, are not inherently interpretable.

Summary and Open Challenges

Time series classification has evolved through several methodological paradigms, ranging from early distance-based techniques to modern ensemble methods and deep representation learning approaches. Distance-based methods provide strong and conceptually simple baselines, but often suffer from poor scalability when applied to long or high-dimensional series. Feature-based approaches, which rely on manually engineered descriptors, offer a degree of interpretability but are highly dependent on careful feature design and domain expertise. Model-based techniques are grounded in solid statistical theory, yet their underlying assumptions can be restrictive and limit their applicability to complex real-world data. In contrast, deep learning methods achieve state-of-the-art accuracy by learning rich hierarchical representations directly from data, but they are typically opaque, computationally demanding, and difficult to interpret. Embedding-based approaches offer a flexible and scalable alternative by transforming time series into fixed-dimensional representations, although they frequently lack transparency and provide limited insight into the decision process.

Despite the wide range of techniques developed for time series classification, the methods that achieve the highest empirical performance are predominantly deep learning or embedding-based models. While effective, these approaches tend to trade interpretability for accuracy, offering limited visibility into the internal mechanisms that drive their predictions. This opacity poses significant challenges in application domains where accountability, safety, and trust are critical, and where understanding the rationale behind a decision is as important as the decision itself.

As a result, the choice of an appropriate classification method remains inherently application-dependent, requiring a careful balance between predictive performance, scalability, and interpretability. Addressing this trade-off remains an open challenge in the field and directly motivates the exploration of neuro-symbolic approaches. By incorporating structured symbolic representations, such as temporal logic, into learning-based models, neuro-symbolic methods aim to combine the expressive power of modern data-driven techniques with transparent, semantically grounded explanations.
However, while neuro-symbolic approaches offer a promising direction, they also raise fundamental questions about what interpretability should mean in practice. Before proposing a concrete framework, it is necessary to examine more carefully how explanation has been defined and operationalised in machine learning. The next chapter therefore shifts focus from classification methods to the broader landscape of explainable and interpretable AI, clarifying the conceptual foundations on which a temporally grounded approach must build.

Chapter 4

The Interpretability Crisis and Explainable AI

The unparalleled performance of deep learning has come at a cost: a profound crisis of interpretability. As opaque models are increasingly deployed in safety-critical domains, their inability to explain their reasoning has evolved from a technical limitation into a fundamental barrier to trust, accountability, and adoption. This tension between power and understanding has given rise to the critical field of Explainable Artificial Intelligence (XAI).

Within this field, two closely related concepts, interpretability and explainability, are often used interchangeably, yet they capture distinct goals [7].

- **Interpretability** refers to the degree to which a human can directly understand the internal mechanics of a model *without* the need for additional explanatory tools. It is an intrinsic property of the model itself (e.g., linear models or decision trees).
- **Explainability** refers to the degree to which the reasoning of a more complex or opaque model can be made understandable through post-hoc techniques, often through the use of external methods that approximate its internal logic (e.g., saliency maps, feature attributions, rule extraction).

Both are essential in high-stakes applications: interpretability ensures that models can be audited at design time, while explainability provides insight into complex models already deployed.

4.1 History of XAI

The concern for model understanding has a clear historical context. The era of *symbolic AI*, or Good Old-Fashioned AI (GOFAI) [86], was characterised by systems built on explicit rules and logical paradigms. These systems, such as decision trees and rule-based classifiers, were *interpretable by design*: a user could trace a line of reasoning back through a chain of rules to understand why a specific conclusion was reached. The shift towards statistical machine learning and, most notably, the rise of deep learning, revived what is often referred to as the *black box* problem. The strength of these modern models, their ability to learn intricate, non-linear patterns from vast amounts of data, is precisely

what makes them opaque. Their complex, high-dimensional internal representations are profoundly difficult for a human to decipher, creating a tension between performance and understanding.

Early machine learning techniques often prioritised interpretability. *Decision trees* [87] provide a transparent, flowchart-like structure where a classification decision can be traced along a path of clear, human-readable conditions. Similarly, *rule-based systems* generate a set of "if-then" rules that are immediately understandable to a domain expert. The trade-off was that these models often lacked the predictive power and flexibility to capture the complexity present in real-world data, leading to the adoption of more powerful but less transparent alternatives. At this stage, the drive for interpretability was largely driven by technical and academic interest in understanding model behaviour.

In the current era, the motivations for Explainable AI (XAI) have moved beyond technical curiosity to become a matter of practical, ethical, and legal necessity. This is driven by the widespread adoption of machine learning in safety-critical domains such as healthcare, finance, and autonomous systems, which has raised profound concerns about the opacity of modern models. Deep learning architectures, while highly accurate, are often criticised for their lack of interpretability, owing to the complexity of their internal representations. This opacity hinders trust, prevents effective debugging, and complicates compliance with emerging regulatory frameworks like the EU's AI Act [88], which mandates transparency and accountability for high-risk AI systems, making explainability a legal requirement rather than just a technical desire. This regulatory pressure, initially sparked by the GDPR and alongside a broader push for *trustworthy AI* [89], has made explainability a prerequisite for deployment. Consequently, stakeholders, including developers, regulators, and end-users, require assurances that models are making decisions for the right reasons, are free from harmful biases, and can be audited. To address these concerns, the research fields of *Explainable Artificial Intelligence* and *Interpretable Machine Learning* have emerged.

4.2 Regulatory Drivers of Explainability: GDPR and EU AI Act

Explainability in AI is not merely a technical desideratum but also a legal imperative especially in the European Union, where two landmark regulations define the landscape. The General Data Protection Regulation (GDPR), which entered into force in 2018, already imposes clear transparency obligations on automated decision-making systems. *Article 22* establishes the "right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her", and *Recital 71* further emphasises that such safeguards should include "the right to obtain an explanation of the decision reached". While legal scholars debate whether this constitutes an explicit "right to explanation" [90], it is widely acknowledged that GDPR obliges data controllers to provide affected individuals with meaningful information about the logic and consequences of automated decisions. This requirement is particularly salient in domains such as lending, hiring, healthcare, and criminal justice, where opaque models can directly impact fundamental rights.

The EU AI Act [88] builds upon this foundation, representing the first comprehensive regulatory framework targeting artificial intelligence as a distinct class of technology. Published in the Official Journal of the EU in July 2024 and entering into force on 1 August 2024 [91], the Act introduces a risk-based framework categorising AI systems into unacceptable, high-risk, limited-risk, and minimal-risk applications.
Provisions are phased in over time [92]: from February 2025, systems deemed "unacceptable risk" (such as social scoring or real-time biometric surveillance in public spaces) are prohibited; from August 2025, governance mechanisms, enforcement structures, and obligations for general-purpose AI (GPAI) come into effect; and by August 2027, full requirements for high-risk AI systems, including conformity assessments, risk management systems, and explainability obligations, will be enforceable. Crucially, Articles 13 and 14 of the AI Act require that high-risk systems be accompanied by transparent documentation, traceability mechanisms, and human–machine interfaces that provide "meaningful information" about their logic, capabilities, and limitations. Violations of these obligations can incur substantial penalties, including fines of up to €35 million or 7% of global turnover, underscoring the centrality of compliance [93].

Taken together, the GDPR and the AI Act position explainability as a cornerstone of *trustworthy AI* within Europe [90, 94, 95]. This regulatory framework elevates the standard for algorithmic transparency: explanations can no longer be ad hoc post-hoc rationalisations, unstable attribution maps, or vague visual cues. Instead, they must be structured, formal, and capable of satisfying both technical and legal scrutiny. For the research community, this means that explainability must be pursued not only as a means of building user trust or debugging models, but also as a compliance obligation tied to verifiability, contestability, and accountability [96]. This shift highlights the need for approaches that provide logically faithful explanations, motivating the use of temporal logics such as Signal Temporal Logic (STL), whose formal semantics align naturally with the requirements of European regulatory frameworks.

4.3 Taxonomy of XAI Methods

The field of Explainable AI (XAI) encompasses a broad range of techniques designed to make machine learning systems more transparent, trustworthy, and accountable. Two methodological paradigms dominate this landscape: the first is *post-hoc explainability*, which produces explanations after a model has been trained, typically for architectures that are not inherently interpretable. The second approach involves designing *interpretable-by-design* models, which integrate transparency directly into their architecture, often trading predictive performance for greater human comprehensibility. Across both paradigms, explanations can target either local or global aspects of the model, where *local explanations* clarify the reasons behind a single prediction, while *global explanations* describe the model's overall behaviour across the dataset.

The following subsections present this taxonomy, with an emphasis on their application and limitations in the context of time series data.

4.3.1 Local vs Global Explanations

Local explanation methods focus on the decision for an individual instance. In time series tasks, these explanations typically highlight influential subsequences, time points, or learned concepts relevant to a specific prediction [97, 98]. Examples include gradient-based saliency maps [99], SHAP values [100], LIME approximations [101], attention-weight visualisations and instance-level prototype matches.
Their primary limitations are instability (small input perturbations can produce different explanations [102]) and weak semantic grounding, especially when domain context is required to interpret highlighted segments [103].

Global explanation methods characterise model behaviour at a system level, capturing patterns or rules that hold across the dataset. In the time series domain, examples include symbolic rules or event primitives approximating decision boundaries [104], global feature importance rankings, prototype sets representing typical class behaviours, learned concept spaces and global concept trends.
These explanations offer high-level insights (e.g., "a rising trend followed by a sharp drop signifies anomaly"), but may oversimplify complex decision functions.

4.3.2 Post-hoc Explainability Methods

Post-hoc XAI methods explain a model after it has been trained and without altering its architecture. They often rely on proxy analyses such as feature attributions, surrogate models, rule extraction and perturbation analysis to rationalise the behaviour of otherwise opaque models. Examples include the aforementioned gradient-based saliency maps, LIME, SHAP, and rule extraction from deep neural networks, and surrogate decision trees and counterfactual explanations.

Despite their widespread use, post-hoc methods face several criticisms. First, they are not guaranteed to faithfully represent the model's true internal logic, leading to what Lipton [36] calls the "interpretability fallacy". Second, explanations can vary across runs, introducing inconsistency. Third, the semantic relevance of highlighted subsequences is often unclear without domain expertise [103]. For example, attribution heatmaps may signal "important" regions in a sensor signal without explaining *why* those regions matter, limiting their utility in safety-critical contexts. In short, post-hoc methods often answer *where* a model focuses, but struggle to explain what or why.

Gradient-based attribution methods. A widely used family of post-hoc explanation techniques relies on gradients of the model output with respect to its inputs or internal representations. The core idea is that the gradient encodes local sensitivity: features (or latent variables) with large partial derivatives have a stronger influence on the prediction. Classical saliency maps compute the gradient of the output score with respect to the input, highlighting regions or time points to which the model is most sensitive [99].

However, raw gradients can be noisy and unstable, particularly in deep networks with saturating nonlinearities. *Integrated Gradients* (IG) [105] was proposed to address these issues by attributing importance through a path integral of gradients between a

baseline input and the actual input. Formally, the attribution for feature i is defined as

$$\mathrm{IG}_i(x) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} \, d\alpha,$$

where x' is a reference baseline and f is the model output. Integrated Gradients satisfies desirable axioms such as completeness and sensitivity, providing more stable and theoretically grounded attributions than single-point gradients.

In the context of time series and concept-based models, gradient-based attribution can be applied not only to raw inputs but also to intermediate representations, such as learned features, concepts, or symbolic robustness scores. This enables explanations that go beyond identifying influential time points, instead quantifying how strongly each intermediate concept contributes to the final decision. Nevertheless, as with other post-hoc methods, gradient-based explanations remain local and may not directly correspond to semantically meaningful reasoning without additional structure or constraints.

4.3.3 Interpretable-by-Design Models

Interpretable-by-design models aim to make the decision process itself transparent by construction. Instead of explaining after the fact, they structure the model around components that can be directly inspected by humans.

Examples include:

- *Linear models and decision trees.* Explicit parameters and rule structures provide a clear linkage between input features and outcomes. However, their limited expressivity makes them ill-suited for high-dimensional or nonlinear time series data.
- *Rule-based and symbolic classifiers.* These rely on handcrafted or automatically extracted symbolic representations (e.g., event primitives [104]) to capture temporal patterns such as peaks, trends, or abrupt changes. These offer strong human interpretability but may require domain expertise for rule definition and validation.
- *Shapelet-based models.* These identify discriminative subsequences whose presence or similarity to the input explains classification decisions [66, 72]. Modern deep shapelet methods learn shapelets jointly with the classifier.
- *Concept-based models.* Decisions are expressed in terms of semantically meaningful concepts, originally in vision (e.g., TCAV [106]) and now extended to time series via prototypes, symbolic abstractions, or human-defined concepts [103, 107]. These support queries like "Is this prediction influenced by the concept of irregular heartbeat?"

Interpretable-by-design models generally provide higher transparency and accountability than post-hoc approaches, but often at the expense of predictive performance, particularly on noisy, high-dimensional, or multivariate time series [37]. Furthermore, the requirement for human-specified concepts or primitives introduces potential bias, as the space of human-interpretable patterns may not fully capture the data's latent structure.

4.4 XAI for Time Series Data

Compared to computer vision and natural language processing, explainability for time series is still a developing field. A central challenge is that raw time series data lack obvious semantics: unlike image patches or words, subsequences of numerical readings may not have inherent meaning without contextual knowledge. This complicates the mapping between algorithmic explanations and human understanding.
Specialised methods have therefore emerged:

- *Attribution-based approaches.* These methods aim to identify which parts of a time series contribute most strongly to a model's prediction, typically by assigning relevance scores to time points, variables, or learned representations. They often adapt gradient-based techniques, such as saliency maps, *Layer-wise Relevance Propagation* (LRP), or Integrated Gradients, as well as attention-based mechanisms [97]. While such approaches can provide fine-grained localisation of influential segments, their outputs may be difficult to interpret or validate without substantial domain expertise, and they do not necessarily guarantee faithful explanations of the model's underlying reasoning.

- *Prototype and clustering methods.* These approaches provide interpretability by associating predictions with representative trajectories or subsequences, effectively grounding explanations in example-based reasoning [103]. This resonates with human reasoning strategies, which often rely on comparison to familiar examples.

- *Rule extraction frameworks.* These methods attempt to build global, symbolic explanations. Temporal rules, logical constraints, or parametrised primitives can capture higher-level dynamics, enabling interpretable reasoning across multiple time scales [104].

- *Hybrid pipelines.* These combine local and global explanations, for example by first using saliency maps to identify important subsequences and then abstracting them into symbolic rules or prototypes [98]. Such hybrid strategies hold promise for bridging low-level attribution with high-level human reasoning.

Recent surveys emphasise that XAI for time series remains young and fragmented, with no widely accepted benchmarks for evaluating explanation quality [103]. Open challenges include defining faithful yet human-meaningful evaluation metrics, handling multivariate and irregularly sampled data, and extending interpretability to real-time and streaming applications. As time series analysis becomes increasingly critical in medicine, finance, and IoT, addressing these gaps will be vital for trustworthy deployment.

The Case for Symbolic Logic

For domains where correctness, accountability, and compliance are paramount, post-hoc explainability may be insufficient [7]. Explanations that rely solely on feature attributions or surrogate models are often non-deterministic, difficult to verify, and lack the guarantees required in high-stakes environments [37]. By contrast, symbolic logic frameworks offer several unique advantages:

- *Mathematical rigour and verifiability.* Logical specifications can be formally reasoned about, verified, and even subjected to automated consistency checks, ensuring that explanations are not only interpretable but also correct by construction.
- *Conciseness and semantic clarity.* Symbolic rules can express temporal phenomena in human-readable forms (e.g., "event A always precedes event B within 10 seconds"), providing immediate interpretability that aligns closely with human reasoning.
- *Regulatory and ethical alignment.* Many application domains are subject to regulations that require transparency and accountability (e.g., GDPR's "right to explanation" [90], see Section 4.2). Logical rules satisfy these demands more directly than statistical attribution maps.
- *Compositionality.* Symbolic logic supports the modular combination of rules, enabling the construction of complex explanations from simpler primitives. This property is crucial for multi-scale or multivariate time series, where different variables may interact in structured ways.

Within this family, *Signal Temporal Logic* (STL) has emerged as a particularly suitable framework for time series. STL provides an expressive language for describing temporal properties over real-valued trajectories [108, 109]. Its dual semantics, Boolean (satisfaction/violation) and quantitative (robustness degree), make it possible to both check whether a property holds and assess how strongly it holds. This flexibility allows STL to encode domain-specific safety constraints (e.g., "temperature must remain below $80°C$ during operation") while also quantifying tolerance margins. Recent work has applied STL to requirement mining, control synthesis, and symbolic classification [110, 111], demonstrating its utility for extracting global rules that characterise system behaviour.

However, these approaches typically remain limited to *global* explanations, offering insights into system-wide dynamics but not into individual classification decisions. The work presented in this thesis extends this paradigm by embedding STL directly into the classification pipeline, enabling both local and global explanations that are logically faithful. This integration bridges the gap between symbolic reasoning and data-driven learning, ensuring that explanations are not only human-readable but also guaranteed to be consistent with the model's decision process.

Seen from this perspective, interpretability in time series is ultimately a question of representation. Capturing temporal dependencies in a form that is both mathematically rigorous and meaningful to domain experts requires an explicit temporal language, such as Signal Temporal Logic. The following chapter introduces its syntax and semantics, laying the groundwork for its later integration into a learning framework.

Chapter 5

Temporal Logic

The ability to formally describe and reason about the behaviour of systems that evolve over time has long been a central challenge in computer science and artificial intelligence. Traditional logical frameworks, such as propositional and first-order logic, are powerful tools for representing static facts and relationships, but they fall short when it comes to expressing how events unfold dynamically. Temporal logic was developed to address this gap, providing a formal language for specifying properties of sequences of states or signals. By incorporating operators that explicitly capture the passage of time, temporal logics make it possible to reason about safety, liveness, causality, and ordering of events in ways that conventional logic cannot [112].

The origins of temporal logic in computer science can be traced to Pnueli's seminal work in the late 1970s, where it was introduced as a specification language for concurrent programs [113]. Since then, temporal logic has become a cornerstone of formal verification, enabling the rigorous analysis of hardware, software, and cyber-physical systems. Over the years, several variants have emerged, each balancing expressivity and computational tractability: *Linear Temporal Logic* (LTL) for reasoning over single infinite traces, *Computation Tree Logic* (CTL) for branching-time models, and *Signal Temporal Logic* (STL) for continuous, real-valued signals.

To illustrate the power of temporal logic, consider a monitoring task in an industrial or medical setting where an anomaly must always be followed by an alert within a strict time bound. Using STL as example, this requirement can be expressed compactly as

$$\mathbf{G}\big(\ \mathit{anomaly} \rightarrow \mathbf{F}_{[0,5]}\ \mathit{alert}\big),$$

which reads: "It must always be the case that if an anomaly occurs, then within five seconds an alert is raised." Such a specification is not only mathematically precise but also intuitively interpretable, showing how temporal logic bridges the gap between formal reasoning and human understanding [114].

Beyond formal verification, temporal logic has recently found new relevance in machine learning. As interest grows in interpretable and trustworthy AI, temporal logics provide a natural bridge between raw sequential data and human-understandable rules. In particular, STL is well suited to time series analysis, offering both Boolean and quantitative semantics that capture not only whether a temporal property holds, but also the degree to which it is satisfied or violated [115]. This makes temporal logic a promising foundation for explainable machine learning methods that demand transparency, accountability, and robustness.

This chapter introduces the main families of temporal logics, with a special focus on STL. We begin by briefly revisiting the foundations of symbolic logic, before moving to modal and temporal logics and their evolution in computer science. We then provide a detailed overview of STL, including its syntax, semantics, and robustness measures, along with illustrative examples. Finally, we discuss how temporal logic has been applied in learning and explainability, highlighting its potential to serve as a rigorous and interpretable framework for AI models operating on time-dependent data.

5.1 Foundations of Symbolic Logic

Symbolic logic forms the backbone of formal reasoning in artificial intelligence and computer science. It provides a precise language for representing knowledge, specifying rules, and deriving valid conclusions. Unlike natural language, which is often ambiguous, symbolic logic ensures that statements can be expressed and interpreted without confusion, making it a cornerstone for automated reasoning systems [116, 117].

At its most basic level, *propositional logic* operates with simple declarative statements, called propositions, that can either be true or false. These propositions are combined using logical connectives such as *and* ($\wedge$), *or* ($\vee$), *not* ($\neg$), and *implies* ($\rightarrow$). For example, the statement "If it rains, then the ground will be wet" can be expressed as $p \rightarrow q$, where p represents "it rains" and q represents "the ground is wet". Propositional logic is widely used in areas like circuit design, rule-based systems, and automated verification [118].

While propositional logic is powerful for reasoning about simple statements, it quickly reaches its limits when more complex relationships are needed. This gap is filled by *first-order logic* (FOL), which extends propositional logic by introducing quantifiers and predicates. Quantifiers such as $\forall$ (for all) and $\exists$ (there exists) allow us to express generalisations and existence claims. For instance, the statement "All humans are mortal" can be represented as $\forall x \, (\mathit{Human}(x) \rightarrow \mathit{Mortal}(x))$. This added expressivity makes FOL a far richer language, capable of describing objects, properties, and relationships between them [119, 120]. As a result, FOL has become central to knowledge representation, automated theorem proving, and database theory.

The importance of symbolic logic in AI goes beyond abstract reasoning. Early AI systems relied heavily on symbolic representations to encode expert knowledge and perform problem solving. While these systems faced scalability challenges, their transparency set them apart from modern black-box approaches. In recent years, symbolic logic has experienced renewed interest through *neuro-symbolic approaches*, which aim to combine the reasoning strengths of logic with the pattern-recognition power of neural networks [117]. This integration promises to deliver AI systems that are both powerful and explainable, aligning well with growing demands for accountability in intelligent technologies.

Through both propositional and first-order systems, symbolic logic provides the formal foundation for representing and reasoning about knowledge in AI. Its ability to express precise rules and relationships makes it particularly valuable for applications requiring clarity and correctness. These foundations will serve as the basis for the temporal and modal logics introduced in the following sections.

5.2 Modal Logics and Their Evolution

While propositional and first-order logic provide powerful tools for reasoning about static facts and relationships, they are less suited for domains where statements must be qualified by modalities such as necessity, possibility, knowledge, or time. To address this, *modal logics* were developed as extensions of classical logic that incorporate modal operators, most notably $\Box$ (necessarily) and $\Diamond$ (possibly). For example, the statement "It is necessary that p" is written as $\Box p$, while "It is possible that q" is expressed as $\Diamond q$ [121, 122].

The roots of modal logic trace back to Aristotle's exploration of necessity and possibility, but its modern formalisation emerged in the early 20th century through the work of C. I. Lewis, who introduced systems of strict implication and the family of modal logics known as S1–S5 [123]. These early systems laid the groundwork for capturing different notions of modality depending on the intended interpretation. For example, S5 assumes that if something is possible, then it is necessarily possible, which aligns with reasoning about possible worlds that are all mutually accessible.

In the mid-20th century, Kripke's relational semantics revolutionised the study of modal logic by introducing *possible worlds semantics*, where the truth of a modal statement is evaluated relative to an accessibility relation between worlds [124]. This framework unified many different modal systems and allowed for precise modelling of concepts such as knowledge, belief, obligation, and temporal change. For instance, *epistemic logic* extends modal logic to reason about what agents know ($K_a p$ for "agent a knows p"), while *deontic* logic applies modal operators to capture obligations and permissions.

Among the most influential descendants of modal logic are the *temporal logics*, which were introduced to reason formally about the ordering of events in dynamic systems. Temporal logic gained prominence in computer science through the seminal work of Pnueli [113], who in 1977 proposed *Linear Temporal Logic* (LTL) as a tool for specifying and verifying concurrent programs. This contribution not only provided a rigorous language for capturing time-dependent properties (such as safety and liveness) but also laid the foundation for model checking, a cornerstone of formal verification [125].

Since then, modal logics have continued to evolve into specialised frameworks tailored to different application domains, including dynamic logic for reasoning about programs, epistemic and doxastic logics for multi-agent systems, and probabilistic modal logics for reasoning under uncertainty [126]. This evolution illustrates how modal logics serve as a bridge between abstract logical theory and practical tools for verification, specification, and knowledge representation.

5.3 Temporal Logics in Computer Science

With the foundational ideas in place, temporal logics became central to computer science as formal tools for specifying and analysing the behaviour of systems that evolve over time. Building on Pnueli's introduction of Linear Temporal Logic, research in this area expanded rapidly, developing a range of operators and semantic frameworks tailored to the needs of concurrent and reactive systems.

Temporal logics differ not only in their syntax, but also in how they model time. The

central distinction is between *linear-time* and *branching-time* frameworks. In linear-time logics such as LTL, time is viewed as a single infinite sequence of states representing one possible execution of a system, and properties are evaluated along that single path. In contrast, branching-time logics such as *Computation Tree Logic* (CTL) interpret time as a tree of possible futures. From any given state, multiple execution paths may unfold, and formulae quantify over these alternatives. This distinction reflects two modelling perspectives: reasoning about individual executions (linear time) versus reasoning about the structure of all possible executions (branching time).

Temporal logics refine propositional reasoning by introducing operators which allow practitioners to express safety, liveness, and ordering constraints over execution traces [127]. These logics underpin many of the algorithms and tools used in formal verification, most notably model checking [125].

LTL extends propositional logic with temporal operators such as:

- $\mathbf{X}\varphi$ ("next φ"): φ holds at the next time step,
- $\mathbf{F}\varphi$ ("eventually φ"): φ holds at some point in the future,
- $\mathbf{G}\varphi$ ("globally φ"): φ holds at all future times,
- $\varphi \mathbf{U} \psi$ ("φ until ψ"): φ holds continuously until ψ becomes true.

For example, the property "every request is eventually followed by a response" can be formally expressed as $\mathbf{G}(\mathit{request} \rightarrow \mathbf{F}\ \mathit{response})$.

In parallel, CTL was developed to reason about branching time structures, where multiple possible futures can unfold from a given state [125]. CTL combines temporal operators with path quantifiers: $\mathbf{A}$ (for all paths) and $\mathbf{E}$ (there exists a path). For instance, the CTL formula $\mathbf{AG}(\mathbf{EF}\ \mathit{recover})$ specifies that from every reachable state, along every possible execution path, there exists a future state where recovery occurs. The distinction between LTL and CTL reflects two different perspectives on time: linear, where the future is a single path, and branching, where multiple futures are possible.

These temporal logics quickly became cornerstones of formal verification, particularly through their integration with *model checking*. Clarke, Emerson, and Sistla demonstrated in 1986 that model checking with temporal logic specifications could be automated efficiently for finite-state systems [125]. This breakthrough led to the widespread adoption of temporal logic in hardware and software verification, where it remains a standard method for specifying *safety* ("something bad never happens") and *liveness* ("something good eventually happens") properties.

Since their introduction, temporal logics have been extended in many directions. Variants such as *CTL** combine the expressiveness of both LTL and CTL, while extensions like *probabilistic temporal logic* capture stochastic behaviours [128]. More recently, temporal logics have been adapted to cyber-physical systems and machine learning, where they serve as formal specification languages for complex time-dependent behaviours [108, 129]. These developments illustrate the central role of temporal logic as a bridge between abstract logical reasoning and practical system verification.

The difference between linear and branching logics is also reflected in their semantic models. LTL formulae are typically interpreted over traces, i.e., infinite sequences of

states generated by a system. In contrast, CTL formulae are evaluated over Kripke structures, which explicitly represent states and transitions between them, thereby capturing the branching structure of computation. From an algorithmic perspective, temporal logic specifications are often translated into automata, enabling automated model checking by verifying language inclusion between system behaviour and the automaton corresponding to the negated specification.

5.4 Signal Temporal Logic

While Linear Temporal Logic and Computation Tree Logic are powerful for reasoning about discrete systems, they are limited when dealing with *real-valued, continuous-time signals*, such as those generated by sensors in cyber-physical systems. To address this, *Signal Temporal Logic* (STL) was introduced as an extension of temporal logic tailored to continuous signals [108]. STL enriches the temporal framework by allowing predicates over real-valued trajectories and by introducing both Boolean and quantitative semantics [129]. This makes STL particularly suited for applications in verification, monitoring, and, increasingly, interpretable machine learning.

5.4.1 Signals and Syntax

Definition 5.4.1 (Signal)**.** A *signal* (or *trajectory*) is a function $x : \mathcal{T} \to D$, where $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ is the time domain, and $D \subseteq \mathbb{R} \cup \{-\infty, +\infty\}$. When $D = \mathbb{B} = \{0, 1\}$, the signal is Boolean; otherwise, it is quantitative (real-valued).

In practice, we typically consider multidimensional signals $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^n$ such that $\boldsymbol{x} = (x_1, \ldots, x_n)$, with each x_i a primary signal representing a component of the system state. Intuitively, a signal encodes the time evolution of a system. Although the time domain $\mathcal{T}$ may be unbounded in theory, in practice signals are observed or simulated over finite intervals.

Definition 5.4.2 (STL Syntax)**.** The syntax of STL formulae φ is given by:

$$\varphi := \top \mid \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathbf{U}_{[a,b]} \varphi_2$$

where $\top$ is the Boolean constant *true*, π is an atomic predicate, $\neg$ and $\wedge$ are standard Boolean connectives, and $\mathbf{U}_{[a,b]}$ is the bounded *until* operator with interval $[a, b] \subseteq \mathbb{R}_{\geq 0}$.

Other operators can be derived from these primitives. Disjunction follows from De Morgan's law:

$$\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2).$$

Similarly, the bounded *eventually* and *always* operators are defined as:

$$\mathbf{F}_{[a,b]}\varphi := \top \, \mathbf{U}_{[a,b]} \varphi, \quad \mathbf{G}_{[a,b]}\varphi := \neg \mathbf{F}_{[a,b]} \neg\varphi.$$

Predicates π are defined over signals by evaluating real-valued functions. Specifically, given a trajectory $\boldsymbol{x}$ and predicate π, we define $\pi(\boldsymbol{x})(t) = (f_\pi(\boldsymbol{x}(t)) \geq 0)$, yielding a Boolean signal. Notably, $f_\pi(\boldsymbol{x}(t))$ is called secondary signal.

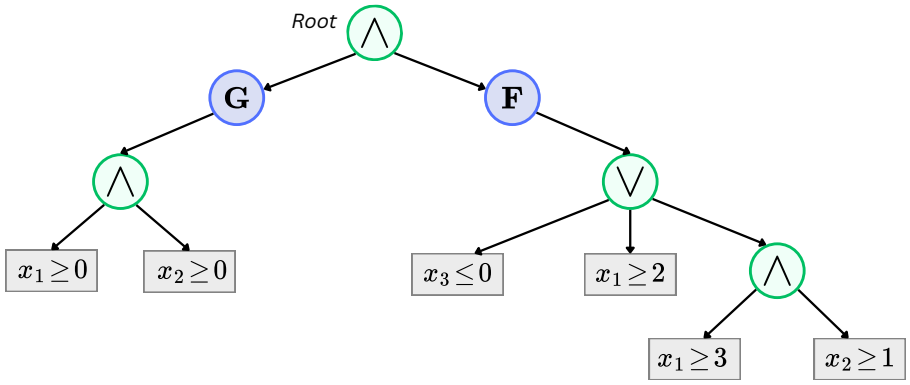


Figure 5.1: Abstract syntax tree representation of the STL formula $\mathbf{G}\big((x_1 \geq 0) \wedge (x_2 \geq 0)\big) \wedge \mathbf{F}\big((x_3 \leq 0) \vee (x_1 \geq 2) \vee \big((x_1 \geq 3) \wedge (x_2 \geq 1)\big)\big)$. The root node corresponds to the top-level conjunction. Internal nodes represent logical and temporal operators (globally $\mathbf{G}$ and eventually $\mathbf{F}$), while leaf nodes correspond to atomic predicates over the signal $\mathbf{x}$. The tree structure makes explicit the compositional nature of STL formulae, enabling recursive evaluation, robustness computation, and structural manipulation.

A convenient way to represent propositional formulae is by means of their Abstract Syntax Tree (AST), where each node represents either an operator (such as conjunction, negation, or a temporal operator) or a leaf predicate. The tree structure mirrors the compositional nature of the logic: complex formulae are built by recursively combining simpler subformulae according to syntactic rules, as illustrated in Figure 5.1. This tree representation is not just a syntactic convenience: it provides a natural basis for algorithmic manipulation, including evaluation, optimisation, and learning. Many STL–based tools exploit the tree structure to recursively compute robustness values, traverse subformulae during monitoring, or generate candidate formulae during specification mining.

5.4.2 Boolean and Quantitative Semantics

Definition 5.4.3 (STL Boolean Semantics)**.** The Boolean satisfaction relation $\vDash$ for an STL formula φ over a signal $\boldsymbol{x}$ at time t is recursively defined as:

$$
\begin{aligned}
&(\boldsymbol{x}, t) \vDash \top \\
&(\boldsymbol{x}, t) \vDash \pi && \leftrightarrow && \pi(\boldsymbol{x}(t)) = \top \\
&(\boldsymbol{x}, t) \vDash \neg\varphi && \leftrightarrow && (\boldsymbol{x}, t) \nvDash \varphi \\
&(\boldsymbol{x}, t) \vDash \varphi_1 \wedge \varphi_2 && \leftrightarrow && (\boldsymbol{x}, t) \vDash \varphi_1 \wedge (\boldsymbol{x}, t) \vDash \varphi_2 \\
&(\boldsymbol{x}, t) \vDash \varphi_1 \, \mathbf{U}_{[a,b]} \, \varphi_2 && \leftrightarrow && \exists t' \in [t+a, t+b] : (\boldsymbol{x}, t') \vDash \varphi_2 \wedge \forall t'' \in [t, t'), (\boldsymbol{x}, t'') \vDash \varphi_1
\end{aligned}
$$

A trace $\boldsymbol{x}$ satisfies φ globally, written $\boldsymbol{x} \vDash \varphi$, if and only if $(\boldsymbol{x}, 0) \vDash \varphi$.

Boolean semantics provide a qualitative yes/no answer about satisfaction. However, for real-valued signals, this can be brittle: small perturbations near a threshold may flip satisfaction, ignoring the degree of violation.

To address this, STL introduces *robust semantics*, which assign a real-valued robustness score to each formula evaluation [129]. This measures not only whether a signal satisfies a property but also by how much.

Definition 5.4.4 (STL Robust Semantics)**.** The *robustness*, or quantitative satisfaction, of φ over signal $\boldsymbol{x}$ at time t, denoted $\rho(\varphi, \boldsymbol{x}, t)$, is recursively defined as:

$$\begin{aligned}
\rho(\pi, \boldsymbol{x}, t) &= f_\pi(\boldsymbol{x}(t)) \\
\rho(\neg\varphi, \boldsymbol{x}, t) &= -\rho(\varphi, \boldsymbol{x}, t) \\
\rho(\varphi_1 \wedge \varphi_2, \boldsymbol{x}, t) &= \min\big(\rho(\varphi_1, \boldsymbol{x}, t), \rho(\varphi_2, \boldsymbol{x}, t)\big) \\
\rho(\varphi_1 \,\mathbf{U}_{[a,b]} \varphi_2, \boldsymbol{x}, t) &= \max_{t' \in [t+a, t+b]} \Big(\min\big(\rho(\varphi_2, \boldsymbol{x}, t'), \min_{t'' \in [t, t')} \rho(\varphi_1, \boldsymbol{x}, t'')\big)\Big).
\end{aligned}$$

Throughout this thesis, unless otherwise stated, all STL formulae are evaluated at the initial time instant. Accordingly, we omit the explicit time argument in the notation and write $\rho(\varphi, \boldsymbol{x})$ as shorthand for $\rho(\varphi, \boldsymbol{x}, 0)$. Similarly, Boolean satisfaction $\boldsymbol{x} \vDash \varphi$ is understood to mean $(\boldsymbol{x}, 0) \vDash \varphi$. This convention simplifies notation without loss of generality and is standard in the analysis of complete signal traces.

Robust semantics are not merely a refinement of Boolean satisfaction, as they are crucial in practical settings. Real-world signals, especially in cyber-physical systems or medical applications, are inherently noisy and subject to measurement uncertainty. Purely Boolean evaluation leads to brittle behaviour, where arbitrarily small perturbations may flip satisfaction. Quantitative robustness provides a notion of stability: if $\rho(\varphi, \boldsymbol{x}) > \varepsilon > 0$, then small perturbations of magnitude less than ϵ will not change the truth value. This continuity property is fundamental for monitoring, optimisation, learning, and controller synthesis, where gradients or margins are required rather than binary outcomes.

The difference between Boolean and quantitative semantics can also be illustrated with a simple predicate such as $\mathbf{x}(t) \geq 0$. Under Boolean semantics, the signal value $\mathbf{x}(t) = 0.001$ satisfies the predicate, while $\mathbf{x}(t) = -0.001$ violates it, even though the two values are nearly indistinguishable. Quantitative semantics, however, assign robustness values $\rho = 0.001$ and $\rho = -0.001$, respectively. The sign determines satisfaction, while the magnitude reflects the distance from the threshold. This additional information becomes essential when reasoning about noisy or uncertain measurements.

5.4.3 STL in Machine Learning

Although Signal Temporal Logic was originally introduced for monitoring and verifying cyber-physical systems, its ability to capture temporal patterns over real-valued signals has increasingly attracted interest within machine learning. The key strength of STL lies in the fact that its formulae are inherently interpretable: they encode properties in a form that can be directly understood by humans, while still being mathematically rigorous. This makes STL an ideal candidate for explainable and trustworthy models, especially in domains such as healthcare, robotics, and industrial monitoring, where both accuracy and accountability are critical.

One important application of STL in machine learning is *specification mining*, the process of automatically extracting logical formulae that describe recurring patterns in data. For instance, from physiological time series such as ECG signals, STL can be used to derive rules like "every QRS spike must be followed by a return to baseline within 0.2 seconds" [130]. These rules are not only interpretable but also provide testable hypotheses for domain experts, bridging the gap between raw data and human understanding. Beyond healthcare, similar approaches have been applied to industrial sensor data, where STL can formalise compliance with operational constraints and detect deviations that correspond to faults or anomalies [131].

Another promising direction is the direct use of STL formulae within classification pipelines. Instead of relying solely on black-box features, STL operators can be combined into logical decision rules that separate classes. For example, a classifier might rely on the property that "an anomaly signal must always be followed by a recovery within ten time steps" to distinguish between safe and unsafe trajectories [111]. Such approaches provide both predictive power and intrinsic interpretability, since each decision is grounded in a temporal rule that can be verified or contested. Moreover, since STL supports both Boolean and quantitative semantics, classifiers can not only determine whether a condition is satisfied, but also measure by how much it is satisfied or violated. This robustness degree adds a further layer of explanatory richness, since it quantifies the margin of safety or severity of violation [129].

Recent developments have also begun to integrate STL into *neuro-symbolic learning*, where logical constraints are embedded within neural architectures. By guiding learning with STL specifications, such models can combine the flexibility of deep learning with the interpretability and verifiability of symbolic reasoning [114]. This hybrid paradigm shows particular promise in safety-critical domains, where it is not sufficient for a model to perform well on average but it must also provably comply with temporal requirements.

However, existing integrations of STL into learning architectures remain fragmented. Many approaches either use logic as an external constraint imposed during training, or focus exclusively on global rule extraction without addressing instance-level explanations. As a result, the interaction between symbolic structure and data-driven optimisation is often partial rather than fully unified.

These limitations suggest that simply incorporating logical constraints is not sufficient. What is needed is a framework in which temporal logic is not an auxiliary component, but an integral part of the representation and decision process. The next chapter therefore examines neuro-symbolic and concept-based models more broadly, analysing how symbolic reasoning can be embedded within learning systems and identifying the structural gaps that remain.

Chapter 6

Neuro-Symbolic AI

The history of artificial intelligence has long been characterised by a fundamental divide between two competing paradigms. *Symbolic AI* (or Good Old-Fashioned AI) [86], which dominated from the 1950s through the 1980s [132, 133], was rooted in logical rules, explicit representations, and structured reasoning. These models offered transparency and rigour but struggled with the ambiguity, noise, and perceptual complexity of real-world environments. In parallel, *connectionism*, exemplified by artificial neural networks, was explored as early as the 1950s [14] but remained marginal until the deep learning revolution of the 2010s [6]. Neural systems demonstrated remarkable capacity in perception, language, and pattern recognition, yet provided little insight into how predictions are produced, lacking the interpretability, compositionality, and logical consistency of their symbolic counterparts.
Early attempts to bridge this divide emerged in the 1980s and 1990s, often combining connectionist architectures with rule-based reasoning [134, 135]. However, these hybrids remained loosely coupled, with limited theoretical foundations and practical impact.

6.1 Modern Neuro-Symbolic AI and Concept-Based Models

The modern revival of neuro-symbolic AI (NeSy) [136] has been driven by advances in deep learning, knowledge representation, and differentiable programming [137]. Contemporary NeSy research is no longer centred on loosely coupled hybrids, but on architectures where neural and symbolic components interact in a deep, bi-directional manner, combining the strengths of neural networks (robust pattern recognition, scalability) with the advantages of symbolic reasoning (interpretability, generalisation, logical consistency).
A prominent example is the family of *concept bottleneck models* [138]. In these architectures, a neural network maps raw input data into a layer of human-understandable concepts. A transparent, often symbolic, model (e.g., a linear classifier or rule set) then produces the final prediction using these concepts. This setup forces the model to reason through an interpretable latent space, enabling explanation, debugging, and human intervention.

The relevance of neuro-symbolic AI to interpretability is direct and profound.

Rather than explaining a black-box model post-hoc, NeSy approaches embed symbolic structure into the architecture itself, providing models whose internal reasoning is more structured, auditable, and aligned with human conceptual frameworks. Consequently, neuro-symbolic AI is positioned not merely as a technical curiosity but as an essential frontier in the development of trustworthy, reliable, and ultimately more intelligent systems [139].

Motivations and Key Principles

The motivation for neuro-symbolic AI lies in addressing fundamental shortcomings of purely neural systems. Deep learning models, despite their success, often require massive datasets, fail to generalise out-of-distribution, and are criticised for their opacity. Symbolic approaches, conversely, provide clear reasoning chains and enable compositionality but struggle with high-dimensional, noisy inputs. The integration of the two aims to achieve:

- *Interpretability and transparency.* The symbolic component provides a foundation that allows the AI to explain its decisions and reasoning processes in a human-understandable way. This directly addresses the black box problem and is a key driver for XAI;
- *Compositional generalisation.* The model can reason over knowledge symbolically, allowing it to generalise from fewer examples than a pure neural net [140];
- *Data efficiency.* Incorporating structured prior knowledge reduces the dependence on large labelled datasets;
- *Trustworthiness and robustness.* Using symbolic constraints enforces consistency with domain knowledge or safety rules;
- *Debuggability and intervention.* The symbolic logic acts as a constraint, providing error handling and users to diagnose and correct model behaviours through semantically meaningful intermediates.

6.2 A Clearer Taxonomy of Neuro-Symbolic Integration

Neuro-symbolic artificial intelligence encompasses a broad family of approaches that seek to combine the expressive power and learning capacity of neural networks with the structure, compositionality, and semantic grounding of symbolic reasoning. Despite this shared objective, existing neuro-symbolic systems differ substantially in how and where symbolic knowledge is integrated within the learning pipeline. To clarify these differences, it is useful to organise neuro-symbolic methods along two complementary dimensions: *when* symbolic components are introduced relative to learning, and *what functional role* they play within the system.

When symbolic knowledge is integrated. A first axis distinguishes neuro-symbolic approaches according to the stage at which symbolic information interacts with the learning process.

Pre-learning integration refers to approaches in which symbolic knowledge is used to construct features, representations, or constraints *before* model training [117, 141]. Examples include the use of logical rules to engineer input features, symbolic abstractions that discretise raw data, or hand-crafted event representations for temporal data. In these cases, the learning algorithm operates entirely in a transformed space, and the symbolic component does not adapt during training.

In-learning integration characterises models in which symbolic and neural components interact *during* learning [142, 143]. This includes architectures where symbolic structures influence gradient-based optimisation, guide representation learning, or form explicit intermediate layers. Hybrid systems in this category often embed symbolic objects into continuous spaces, impose differentiable logical constraints, or structure the model around interpretable intermediate representations. Because symbolic information is present throughout training, these approaches allow mutual adaptation between neural representations and symbolic structure.

Post-learning integration encompasses methods in which symbolic reasoning is applied *after* a neural model has been trained [135, 144]. Typical examples include rule extraction from trained networks, symbolic approximation of decision boundaries, or formal verification and constraint checking applied to a fixed model. While these approaches can provide valuable insights or guarantees, the symbolic component does not influence the learned representation itself.

The functional role of symbols. A second, orthogonal axis distinguishes neuro-symbolic methods by the role played by symbolic knowledge within the system.

In some approaches, symbols act as *constraints* or *priors*, restricting the space of admissible solutions [145, 146]. Logical rules may enforce consistency, encode physical laws, or rule out implausible predictions. Here, symbols shape learning indirectly, by penalising violations or guiding optimisation, rather than serving as explicit representational units.

Other methods use symbols as *targets* or *auxiliary supervision*. In these settings, neural networks are trained to predict symbolic outputs (e.g., logical labels, events, or predicates), which may then be composed or reasoned over by a separate symbolic module. The symbolic layer operates downstream of perception, often assuming that the predicted symbols are correct.

A third and increasingly influential role is that of symbols as *intermediate representations* [106, 138]. In this case, symbolic entities form an explicit bottleneck or latent space through which information must pass in order to reach the final prediction. These representations are typically human-interpretable and semantically meaningful, allowing decisions to be expressed directly in terms of symbolic concepts rather than opaque latent features. Concept-based models belong to this category and are discussed in detail in the following section.

Finally, symbols may be used purely as *explanatory artefacts*, providing post-hoc interpretations of a model's behaviour without constraining or structuring the learning process itself. While such explanations can be useful for inspection or debugging, they

do not guarantee faithfulness to the model's internal reasoning.

6.3 Architectures of Neuro-Symbolic Systems

Different architectural strategies have been developed for integrating symbolic reasoning with neural learning. Some approaches inject symbolic knowledge directly into neural training as constraints or regularisers, embedding logical structure into the optimisation objective. Others adopt a modular design in which perception and reasoning are separated: neural components extract structured representations from raw data, and symbolic reasoners operate on these intermediate abstractions [137]. A third line of work develops differentiable logic frameworks, where symbolic rules are relaxed into smooth operators that can be optimised end-to-end [147]. Reinforcement learning has likewise been extended with neuro-symbolic constraints, enabling agents to incorporate logical rules while learning policies in complex environments.

Despite this variety of architectures, a central obstacle persists: the *representational gap* [148]. Neural networks operate in continuous, high-dimensional vector spaces, with internal states encoded as tensors (multidimensional arrays of floating-point numbers). Symbolic AI, by contrast, relies on discrete, explicit structures such as logical rules (e.g., $A \wedge B \Rightarrow C$), knowledge graphs, and symbolic predicates. This mismatch leads directly the *gradient problem*: neural networks learn via backpropagation, and therefore require each computational component to be differentiable so that gradients can be propagated via the chain rule.
Symbolic logic, however, is inherently non-differentiable: a logical statement is either $true$ or $false$, and therefore yields no gradient to compute. This incompatibility forces system designers into a trade-off between softening symbolic rules to enable optimisation at the cost of logical guarantees [149], and maintaining discrete symbolic reasoning, which preserves rigour but complicates joint learning [150].

Importantly, while symbolic components are traditionally non-differentiable, the use of quantitative semantics and differentiable surrogate representations enables gradient-based analysis of the decision process.

6.3.1 Concept-Based Models

A particularly influential line of research within interpretable neuro-symbolic AI is *concept-based modelling*. These models introduce human-understandable concepts as intermediate variables between raw input and final prediction. The most prominent example is the *concept bottleneck model* (CBM) [138], in which a neural network first predicts a set of predefined concepts and then uses those concepts for classification. Figure 6.1 illustrates the pipeline: an image of a bird is mapped to semantically meaningful attributes (e.g., "wing colour", "beak length"), which act as the interpretability bottleneck before the final prediction.
This design offers three advantages. First, it provides *transparency*: predictions can be explicitly traced to concept activations. Second, it enables *intervention*: users can adjust concept predictions to see how outputs change, helping identify spurious correlations and correcting errors. Third, the structure aligns naturally with symbolic reasoning

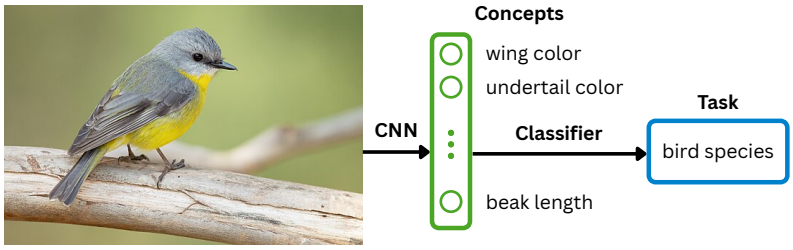


Figure 6.1: Concept bottleneck model architecture. A CNN extracts human-interpretable concepts (e.g., wing colour, undertail colour, beak length) from raw input, which then serve as an interpretability bottleneck for downstream classification. This design enables transparent prediction paths and human intervention at the concept level.

frameworks, since concepts serve as discrete predicates that can be combined into rules or constraints.

Beyond CBMs, a variety of concept-based approaches has emerged. Interactive concept bottleneck models allow human users to provide concept labels during inference to refine predictions [151]. This turns the bottleneck into a collaborative interface rather than a static module. Other works incorporate concepts into memory-based architectures. Concept-based memory reasoning [152] augments neural networks with symbolic rule evaluation over stored concept representations, enabling verifiable decision paths and stronger guarantees. A complementary direction moves away from predefined concept sets entirely. *Explanation bottleneck models* generate natural language explanations that act as the bottleneck for prediction [153]. Instead of attributes, the intermediate layer consists of textual rationales, providing human-aligned reasoning while remaining trainable end-to-end. Recent surveys highlight concept-based reasoning as a middle ground between uninterpretable neural models and brittle purely symbolic systems [154]. The shared theme is to restructure neural computation around semantic entities humans use to describe and reason about phenomena.

Positioning concept-based approaches. Viewed through the taxonomy introduced above, concept-based models occupy a distinctive position within the neuro-symbolic landscape. They are characterised by *in-learning* integration, where symbolic concepts are embedded directly into the model architecture, and by the use of symbols as *intermediate representations* rather than mere constraints or post-hoc explanations [155, 156]. This design choice enables interpretability by construction: predictions are mediated by concept activations that retain semantic meaning throughout training and inference.

This perspective highlights a key distinction between concept-based neuro-symbolic models and approaches that rely on symbolic reasoning only before or after learning. Rather than explaining a black-box model retrospectively, concept-based architectures seek to align learning, reasoning, and explanation within a single, unified framework. The implications of this design choice, particularly for sequential and temporal data, are examined in the next section.

Architecture and Limits

While the literature presents CBMs primarily as interpretable pipelines, their defining characteristic is architectural. CBMs explicitly factorise prediction through a concept space. Given an input $x \in \mathcal{X}$ and label $y \in \mathcal{Y}$, prediction is decomposed as

$$x \xrightarrow{f_{\text{enc}}} \mathbf{c}(x) \xrightarrow{f_{\text{clf}}} \hat{y},$$

where $\mathbf{c}(x) = (c_1(x), \ldots, c_C(x))$ denotes concept activations. This *concept bottleneck* enforces that all information relevant for prediction is mediated by semantically meaningful variables.

Concepts may be *hard*, taking binary values that align with logical predicates, or *soft*, taking continuous values that encode degrees of presence or confidence. Soft concepts are particularly important in noisy or continuous domains, where crisp symbolic boundaries are inappropriate. In either case, the bottleneck ensures that explanations arise intrinsically from the model's computation rather than from post-hoc analysis.

Despite these advantages, existing CBMs face several limitations. Many rely on manually defined or annotated concept sets, which may be incomplete or expensive to obtain. Concept prediction modules can suffer from entanglement or information leakage, undermining the intended separation between representation and decision. Moreover, most CBMs have been developed for static data modalities, where concepts correspond to spatial attributes or object properties.

Extending CBMs to sequential and temporal data introduces additional challenges. Temporal concepts often involve ordering, duration, and logical composition of events, which are difficult to express using simple attribute-based concepts. Existing approaches typically resort to subsequences, prototypes, or latent embeddings, which provide limited semantic grounding. These limitations motivate the exploration of concept representations that are formally defined and naturally suited to temporal reasoning.

Explainability and Trustworthiness

One of the central motivations behind concept-based models in neuro-symbolic AI is improved explainability. Because predictions must pass through an interpretable layer, CBMs allow domain experts to inspect whether the model's internal reasoning matches scientific, clinical, or operational expectations. For example, in medical imaging, a pneumonia classifier could reveal that its decision relies on interpretable radiological features such as "lung opacity" or "pleural effusion". This makes it easier to validate (or contest) the model's behaviour and detect shortcut learning. In high-stakes settings, concept-level explanations often better align with domain ontology than pixel-level heatmaps.

Neuro-symbolic extensions further strengthen these guarantees. When concept activations are treated as symbolic predicates, verification tools can ensure that decisions satisfy domain-specific rules before deployment [152]. This connects concept-based AI with broader responsible AI goals, including robustness, fairness, and accountability. For instance, models can be checked to ensure they do not rely on prohibited concepts (e.g., demographic attributes) or violate safety constraints.

Applications and challenges

Concept-based and neuro-symbolic models have been successfully applied across a range of domains:

- *Healthcare.* CBMs support interpretable diagnosis by mapping raw signals (e.g., images, ECG, or lab values) to medically grounded concepts. Symbolic constraints can enforce clinical guidelines or consistency rules.
- *Robotics.* Neuro-symbolic planning integrates learned perceptual concepts with symbolic task or motion specifications, enabling robots to reason about goals, constraints, and safety while retaining flexible perception.
- *Natural language understanding.* Concept-based abstractions help neural models exhibit compositional generalisation, consistency, and improved reasoning over relational structures.
- *Scientific discovery.* Hybrid models combine neural perception with symbolic rule extraction, enabling identification of interpretable relationships or governing laws from complex data.

Despite substantial progress, several open challenges remain. First, concept-based models often face a predictive performance trade-off. By constraining the model to operate through concepts, performance may drop when the predefined concept set is incomplete, ill-chosen, or too coarse. Recent work seeks to mitigate this through learned concept spaces or hybrid bottlenecks. Second, scalability is a persistent issue. Neuro-symbolic reasoning becomes computationally demanding when dealing with large symbolic knowledge bases or long-horizon reasoning tasks. Efficient neurosymbolic architectures, pruning methods, and differentiable approximations remain active research directions. Third, integration frameworks remain fragmented. Different communities (XAI, knowledge representation, deep learning, cognitive science) propose heterogeneous architectures that lack a unifying theory. This complicates benchmarking and comparison across approaches.
Finally, there is increasing interest in *meta-cognition*: enabling models not only to reason about data but to reason about the structure, quality, and uncertainty of their own reasoning processes [147]. This direction is especially relevant for neuro-symbolic systems, where both concepts and rules may require explicit confidence estimates or provenance tracking.

Taken together, these challenges indicate that the integration of symbolic reasoning and statistical learning remains structurally unresolved. Existing approaches demonstrate feasibility, but a coherent and principled framework that reconciles expressiveness, scalability, and interpretability is still lacking.

Chapter 7

Summary and Identified Research Gaps

The preceding chapters have outlined the foundations that support this thesis: time series classification, explainable and interpretable machine learning, temporal logics for reasoning over dynamic data, and the emerging neuro-symbolic paradigm centred on concept-based models. Together, these domains reveal both substantial progress and persistent limitations that continue to restrict the transparency, reliability, and practical deployment of learning systems, particularly in safety-critical applications.

Time series classification has evolved from distance-based and feature-engineered approaches to modern deep learning architectures capable of modelling complex temporal dependencies. Although these models often achieve excellent predictive performance, they remain largely opaque and computationally demanding. Moreover, post-hoc explanation techniques, now pervasive in practice, are frequently unstable, weakly faithful to the underlying model, or misaligned with domain semantics. Intrinsically interpretable models, including concept-based approaches, alleviate some of these issues but commonly rely on predefined, rigid concept vocabularies or lack the expressiveness needed to describe temporal phenomena.

Temporal logics such as Signal Temporal Logic offer a principled, expressive language for reasoning about time. Their Boolean and quantitative semantics make them promising tools for interpretable machine learning, yet integration with modern learning pipelines has been limited. Existing methods typically rely on handcrafted formulae, limited search spaces, or post-hoc rule extraction, none of which scale well or guarantee alignment with the classifier's internal reasoning. Neuro-symbolic AI aims to bridge these gaps by combining neural perception with symbolic reasoning, but most work focuses on static data or high-level tasks, leaving time series analysis relatively underexplored.

These considerations point to a set of concrete, interconnected research gaps:

1. *A lack of unified frameworks that jointly optimise predictive accuracy, interpretability, and computational efficiency for time series.* Existing models tend to prioritise one or two of these properties at the expense of the others, preventing widespread adoption in real-world settings.

2. *A shortage of explanation methods with formal grounding.* Post-hoc explanations

often lack guarantees of faithfulness or stability, while symbolic approaches are either manually designed or too restrictive to reflect the classifier's reasoning.

3. *Limited integration of STL within contemporary learning architectures.* Although STL is expressive and semantically rich, current methods seldom use it as a learnable explanatory layer or as part of an end-to-end training pipeline.

4. *Concept-based models with limited temporal expressivity.* Existing concept vocabularies are typically static, predefined, and poorly matched to the structure of time series, making them unsuitable for capturing temporal abstractions.

5. *Insufficient neuro-symbolic approaches tailored to sequential data.* Despite growing interest in neuro-symbolic AI, most frameworks target static vision or language tasks, leaving a gap in approaches that unify temporal logic, learned concepts, and data-driven classification.

The STELLE framework introduced in this thesis is designed precisely to address these gaps. By embedding STL semantics directly into the learning process through robustness-based embeddings, STELLE provides a unified model that learns temporal concepts aligned with discriminative prototypes. These concepts form an interpretable latent space through which all predictions must pass, ensuring that explanations are not merely post-hoc approximations but intrinsic components of the model's reasoning. In linking STL and neuro-symbolic interpretability into a coherent architecture, STELLE fills a crucial gap in the literature: it demonstrates that time series classifiers can be simultaneously accurate, efficient, and transparently grounded in formal temporal logic. This contribution advances the broader goal of trustworthy AI for temporal data, offering a practical and semantically principled alternative to the black-box models that dominate the field.

Part II

Core Contribution

Chapter 8

STELLE

The preceding chapters examined the evolution of machine learning, the challenges of time series classification, and the limits of current explainability and neuro-symbolic approaches. Together, they highlighted the absence of a framework in which temporal logic is fully integrated into the learning process.

Machine learning for time series classification has advanced considerably in recent years, fuelled by increasingly expressive neural architectures capable of modelling complex temporal dependencies. Despite their success, these models typically offer limited insight into the temporal patterns that drive their decisions. Post-hoc explanation techniques such as saliency maps or perturbation-based analyses attempt to address this limitation, yet they often lack semantic clarity and may fail to reflect the true internal reasoning of the model. In domains where safety, accountability, and domain expertise play a central role, such opacity poses a significant barrier to adoption.

This tension between accuracy and interpretability motivates the development of frameworks that integrate symbolic reasoning directly into the learning process. A particularly promising direction emerges from the use of Signal Temporal Logic (STL), a formal language that allows one to express rich temporal specifications over real-valued signals, introduced in Section 5.4. STL provides both Boolean and quantitative semantics: the latter measures not only whether a temporal condition is satisfied, but also the degree to which it is satisfied. This quantitative view aligns naturally with machine learning, where continuous scores and differentiable computations are essential.

However, despite its relevance, STL has seen limited integration within modern time series learning pipelines. Existing approaches grounded in temporal logic often focus on property mining, verification, or task-specific rule extraction. These methods typically operate as post-processing stages or require substantial domain knowledge to design meaningful STL specifications. Conversely, deep learning models rarely exploit structured temporal concepts during training, and instead rely on latent representations whose relation to temporal rules remains opaque.

The STELLE framework introduced in this thesis addresses this gap. It is built upon the idea that *interpretable concepts should be temporal, symbolic, and grounded in the semantics of the data.* To this end, STELLE constructs a large and diverse set of candidate STL formulae, interprets them as human-readable temporal concepts, and embeds both formulae and trajectories into a shared space where similarity is defined through STL robustness. This enables a classifier that is inherently concept-based, in the spirit of prototype and concept learning, yet explicitly grounded in formal temporal logic.

The goals of this part of the thesis are threefold:

1. to motivate the integration of STL semantics into concept-based learning;
2. to provide a comprehensive and detailed description of the STELLE framework, including its architecture, theoretical underpinnings, and practical implementation;
3. to evaluate the quality of the resulting classifiers and explanations through extensive experiments, including comparisons with STL-based baselines, ablation studies, and sensitivity analyses.

Compared to previous work, STELLE contributes a unified approach where classification and explanation emerge from the same semantic space. Local explanations identify the most influential temporal concepts underlying each decision, while global explanations capture the common temporal signatures of each class, both in the form of STL rules. These explanations are symbolic, simplified, and directly interpretable, offering a level of transparency uncommon in contemporary time series classifiers.

Running Example. To clarify the mechanics of the framework, we introduce a simple univariate time series that will be reused throughout this chapter. Consider a trajectory $\mathbf{x} \in \mathbb{R}^{50}$, indexed over discrete time steps $t = 0, \ldots, 49$. The signal remains close to zero for most of its duration and exhibits a pronounced peak around time step $t = 20$, as shown in Figure 8.1.

We consider two illustrative STL formulae:

$$\varphi_1 = \mathbf{F}_{[15,25]}(x > 2)$$

which requires the signal to exceed the threshold value 2 at least once within the interval $[15, 25]$, and

$$\varphi_2 = \mathbf{G}_{[0,50]}(x < 1.5)$$

which enforces that the signal remains globally below 1.5 throughout its evolution. Intuitively, the trajectory strongly satisfies φ_1, due to the peak in the highlighted interval, and violates φ_2. These two simple concepts will serve as a concrete reference

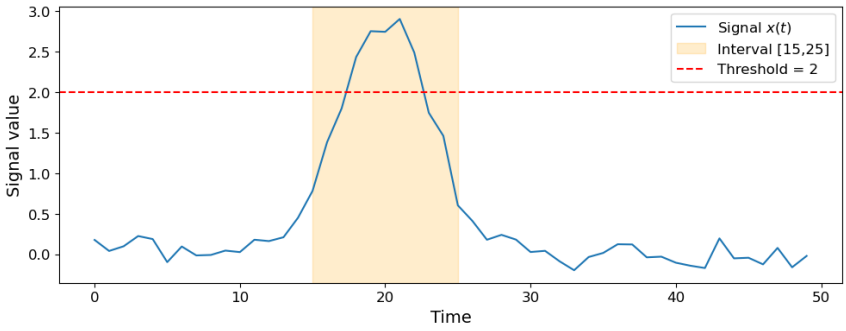


Figure 8.1: Example univariate time series used as a running illustration in Chapter 8. The shaded region corresponds to the interval $[15, 25]$, and the dashed horizontal line denotes the threshold value 2. The peak within the highlighted interval satisfies the temporal property $\mathbf{F}_{[15,25]}(x > 2)$.

when illustrating robustness computation, kernel embedding, discriminability, and explanation extraction.

st

8.1 Positioning STELLE within the Literature

Time Series Classification (TSC) lies at the intersection of pattern recognition, dynamical systems analysis, and modern machine learning. The task of learning a mapping between multivariate temporal trajectories and categorical outcomes is central to domains such as healthcare monitoring, finance, and human activity recognition. As previously discussed in Section 3, over the last decade the field has evolved from distance-based similarity measures and handcrafted features to deep architectures capable of capturing complex temporal dependencies. However, this performance leap has come at the expense of *interpretability*: the reasoning behind predictions often remains opaque.

The STELLE framework (Symbolic Temporal Explainable Latent Logic Embedding) is conceived at the confluence of three active research streams: (i) Explainable AI for time series, (ii) neural-symbolic reasoning and the learning of temporal logic formulae, and (iii) concept-based and interpretable representation learning. This chapter reviews these literatures to clarify how STELLE extends and differentiates itself from prior work, highlighting its contribution as a logic-grounded, concept-based model for interpretable time series learning.

8.1.1 Time Series Classification

Traditional approaches to TSC rely on comparing entire time series through distance-based methods (e.g., Dynamic Time Warping, DTW), computing summary statistics to use as fixed-length feature vectors, or leveraging characteristic subsequences, known as *shapelets*, to represent discriminative temporal patterns [157, 158]. These methods typically produce interpretable models, as the reasoning can be traced back to specific features or subsequences. However, they often struggle to capture multivariate interactions, nonlinear dependencies, and long-term temporal structure.

Deep learning approaches have since become state-of-the-art for TSC [159]. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) learn complex temporal dependencies directly from data, while hybrid architectures such as Temporal Convolutional Networks (TCNs) and attention-based models extend their temporal receptive field. Transformer architectures [32] further advanced this direction by using self-attention to model long-range dependencies without recurrence. This paradigm has culminated in the development of *foundation models* for time series [160], capable of learning general-purpose temporal representations across heterogeneous datasets and downstream tasks.

Despite their flexibility and predictive power, these models are often opaque. They prioritise performance over interpretability, offering limited insight into which temporal features drive predictions. This trade-off has motivated a growing body of research in explainable artificial intelligence (XAI) for temporal data.

8.1.2 Explainable AI for Time Series Classification

Explainable AI for time series largely adapts techniques developed in other data domains, including computer vision and tabular data, to sequential settings [161, 162]. Attribution-based methods are among the most widely used: they aim to identify time points or subsequences that most influenced a model's prediction, often through gradient-based or perturbation-based analysis. Perturbation approaches, for instance, systematically alter parts of the input sequence and observe changes in model output, while gradient-based methods compute relevance scores using backpropagation.

Attention-based explanations, by contrast, incorporate attention mechanisms directly into the classifier [163, 164, 165]. The attention weights can then be interpreted as measures of temporal relevance, highlighting the portions of the sequence that contributed most to the prediction. Recent extensions target multivariate settings, designing architectures that simultaneously learn to attend to important time intervals and relevant variables [166, 167].

Beyond temporal attribution, other interpretability strategies have emerged. Prototype-based models provide interpretable representations by identifying class-representative trajectories, while symbolic and counterfactual approaches focus on generating human-readable explanations. For instance, Baldán and Benítez [168] propose prototype-based latent representations of trajectories, and Refoyo and Luengo [169] introduce sparse, subsequence-level counterfactual explanations indicating minimal changes required to alter a classification. Ensemble-based frameworks have also been used to identify robust, medically relevant features in multivariate TSC tasks [170]. de Graaff et al. [171] present a functional decomposition method aligning latent behaviour with meaningful subsequence dynamics, and symbolic frameworks such as EMeriTAte+DF [172] demonstrate that hybrid numeric-event reasoning can enhance both verifiability and explainability.

Toolkit developments such as `tsinterpret` [173] and systematic evaluations of perturbation robustness [174] highlight a growing interest in benchmarking and standardising time series XAI. However, recent surveys [175, 176, 177] stress the lack of *time-series-specific* frameworks that combine interpretability with formal guarantees. While the above methods enhance the transparency of deep models, they generally produce *post-hoc* and non-verifiable explanations.

In contrast, STELLE embeds interpretability within its architecture by grounding explanations in the formal semantics of Signal Temporal Logic. STL provides a well-defined language for expressing temporal relationships (e.g., "whenever variable x exceeds threshold θ_1, variable y falls below θ_2 within Δt seconds"). By representing explainable concepts as STL formulae, STELLE ensures that explanations are not only interpretable but also logically verifiable and consistent across model evaluations.

8.1.3 Learning Temporal Logic Formulae as Time Series Classifiers

Learning temporal logic formulae directly from data has received increasing attention as a means of bridging symbolic reasoning and machine learning. Pioneer works [110, 111, 178] cast the TSC problem as a data-driven requirement mining task, in which classifiers are constructed from STL properties mined from labelled trajectories. Related

symbolic approaches, such as those by Kadous [179], explored parameterised event primitives to extract rule-based classifiers from time series, anticipating more recent advances in symbolic temporal reasoning.

Recent efforts in neural-symbolic (NeSy) modelling have advanced this idea in two main directions. A first line of research combines decision tree frameworks with STL-based decision nodes, where activation functions approximate robust semantics and neural networks are used to guide tree splits [180, 181]. A second line translates the syntax tree of a temporal logic formula directly into a neural network, where neurons correspond to logical operators and activation functions are designed to emulate robustness-based truth values [182, 183].

These models achieve interpretability by explicitly encoding logical structure but typically yield *global* explanations: a single STL formula or decision tree that describes an entire class. In contrast, STELLE natively computes *local* explanations, identifying logic-based concepts most relevant to a given instance. From these local explanations, global summaries can be naturally inferred, enabling a multi-scale interpretability that balances precision with generality. Furthermore, STELLE evaluates a fixed set of human-interpretable STL formulae directly on the input trajectories, mapping both STL concepts and trajectories into a shared latent space where relevance weights can be assigned. This design encourages diversity and conciseness in the explanations, while maintaining a strict logical grounding.

8.1.4 Concept-Based Models for Time Series Data

Concept-based interpretability aims to explain model decisions in terms of human-understandable, semantically coherent concepts rather than raw features. In the time series domain, such models are comparatively rare. Jastrzebska et al. [184] use concepts obtained by clustering raw time series, sometimes integrating domain knowledge, to perform interpretable classification and data analysis. Prototype-based approaches [185, 186, 187] similarly identify representative trajectories that serve as conceptual anchors for each class.

Other methods extend the Concept Activation Vector (CAV) framework to TSC [188], using human-defined concepts to probe network sensitivity. Symbolic abstraction through interpretable rule sets and logical primitives [179] has also been explored, but typically relies on handcrafted templates or predefined event grammars, limiting their flexibility.

To the best of our knowledge, STELLE is the first approach to propose a neural-symbolic, concept-based architecture for TSC in which temporal logic concepts are automatically inferred from data, without human intervention. By mapping STL formulae and trajectories into a shared embedding space, STELLE enables concept-level reasoning that is both data-driven and formally interpretable.

8.1.5 Learning Meaningful Representations of Time Series

The quest for meaningful and compact representations of time series has been central to machine learning for decades. Classical methods such as DTW combined with dimensionality reduction [189] or shapelet-based transformations [66, 190] offered interpretable yet limited embeddings. More recently, contrastive and self-supervised

techniques like TS2Vec [191] and LETS-C [192] have achieved universal, task-agnostic representations by focusing on predictive generalisation. However, these embeddings prioritise utility over interpretability, making them unsuitable for domains requiring human-understandable insights.

Emerging directions in self-explainable and graph-based temporal models [193], and verbalised spatio-temporal explanations [194], suggest a growing interest in incorporating interpretability into the representation learning process. In this context, STELLE proposes a concept-aligned embedding space defined by the evaluation of interpretable STL formulae. By structuring the embedding space around logical semantics, it unifies classification and explanation within a single, interpretable representation framework.

8.1.6 Synthesis

The evolution of time series learning reveals a clear trajectory: from interpretable but rigid symbolic models, to flexible yet opaque neural networks, and now toward hybrid approaches combining both. Despite numerous advances, existing frameworks tend to fall short in one or more dimensions: they are either post-hoc and non-verifiable, limited to global symbolic descriptions, or dependent on handcrafted abstractions.

STELLE bridges these divides by unifying neural representation learning with symbolic temporal logic reasoning. It grounds explainability in the formal semantics of STL, ensuring verifiability, while employing attention-based neural embeddings to capture contextual relevance and diversity. The resulting framework delivers both local and global interpretability, reconciling data-driven flexibility with logical rigour. In doing so, STELLE defines a new paradigm for interpretable, verifiable, and concept-based time series learning.

8.2 STL as Concept Language and Concept Set Construction

Signal Temporal Logic provides a formal, interpretable, and expressive language for describing temporal properties of real-valued signals [108, 109]. Its quantitative semantics enable a principled measure of how strongly a temporal condition holds, making STL particularly well suited for integration with machine learning models that operate on continuous scores. In the context of concept-based learning, STL serves as a semantically grounded space of human-interpretable temporal concepts represented by well-structured logical formulae. This chapter presents STL as the concept language adopted by STELLE, and describes in detail how the concept set is constructed through a probabilistic grammar, redundancy-aware selection mechanisms, and dataset-specific scaling procedures.

8.2.1 STL as a Concept Space

In STELLE, each STL formula φ constitutes an interpretable temporal concept. The robustness value $\rho(\varphi, \mathbf{x})$, as defined in Section 5.4, is interpreted as the response of

concept φ when the model processes an input trajectory $\mathbf{x}$. This viewpoint offers several key advantages.

1. *Semantic grounding.* Each concept has an explicitly defined temporal meaning grounded in the formal syntax and semantics of STL. The meaning of a formula is invariant across datasets, feature scalings, or architectures.

2. *Continuity.* The quantitative semantics provide a smooth, real-valued measure of satisfaction that is suitable for gradient-based optimisation and compatible with neural learning objectives.

3. *Compositionality.* STL formulae can capture complex temporal patterns through systematic composition of atomic predicates, Boolean operators, and temporal operators, enabling the concept set to span a broad range of interpretable structures.

4. *Temporal sensitivity.* STL captures timing, ordering, persistence, and magnitude in a single formalism. Concept activations therefore encode temporal relations rather than static or purely local features.

Embedding both trajectories and formulae into a shared robustness-based space yields a concept-learning process that is intrinsically interpretable: the model does not discover latent representations but instead reasons directly using explicit temporal structures.

From a structural perspective, this construction relies on minimal assumptions about the input signals. Trajectories are assumed to be real-valued and discretely time-indexed, so that STL robustness can be evaluated over finite intervals. No assumptions are made regarding stationarity, smoothness, periodicity, linear dynamics, or parametric generative models. In particular, the framework does not presuppose any specific distribution over trajectories, nor does it require differentiability or continuity of the signals beyond what is needed for numerical robustness computation. Temporal properties are evaluated pointwise and over bounded intervals, making the approach agnostic to the underlying data-generating mechanism. As a consequence, STELLE remains applicable across heterogeneous domains, provided that signals admit quantitative STL semantics.

Running example. To make this interpretation concrete, consider the running example introduced at the beginning of this chapter (Figure 8.1). Let

$$\varphi_1 = \mathbf{F}_{[15,25]}(x > 2) \qquad \varphi_2 = \mathbf{G}_{[0,50]}(x < 1.5).$$

The robustness value $\rho(\varphi_1, \mathbf{x})$ is positive, since the signal exceeds the threshold within the specified interval, and its magnitude reflects the margin by which this exceedance occurs. Conversely,$\rho(\varphi_2, \mathbf{x})$ is negative, as the trajectory violates the global amplitude constraint.

This simple example illustrates how STL formulae operate as temporal concepts: each robustness value quantifies the degree to which a semantically meaningful temporal property is exhibited by the trajectory. Concept activations are therefore not abstract latent features, but structured evaluations of explicitly defined behavioural conditions.

Suitability of STL for Interpretable Time Series Learning

STL is particularly suitable for interpretable time series learning due to:

- *Transparency.* Each formula presents a clear, human-readable temporal condition.
- *Faithfulness.* Robustness values are tightly coupled to the logical semantics of STL, ensuring that the model's internal signals correspond to meaningful behavioural patterns.
- *Expressiveness.* Temporal operators allow formulae to specify timing, duration, recurrence, ordering, and persistence, which are difficult to represent with shape-based or threshold-based features.
- *Modularity.* Formulae can be simplified, combined, or transformed without losing interpretability, and their structure is directly reflected in their semantics.

However, challenges arise when sampling large sets of formulae or evaluating robustness for complex trees. STELLE mitigates these limitations through principled grammar constraints, redundancy-aware concept selection, and efficient robustness computation.

8.2.2 Concept Set Construction

The STL concept set must satisfy several properties that ensure both interpretability and usefulness for learning:

- *Interpretability.* Formulae are kept shallow, with bounded temporal operators and low syntactic complexity to support human understanding.
- *Behavioural diversity.* The concept set should span a rich variety of temporal patterns, avoiding excessive redundancy whilst maintaining discriminative power.
- *Coverage of temporal patterns.* Concepts should collectively represent the main variability modes present in the dataset, capturing the characteristic behaviours of different classes.
- *Computational tractability.* Formulae must remain pliant to efficient robustness computation, avoiding excessively deep nesting or overly large temporal intervals.

These principles guide the construction pipeline described below, which balances expressiveness against interpretability through grammar-based generation and redundancy-aware selection.

Probabilistic Grammar and Sampling

The STL concept set is generated using a probabilistic grammar inspired by the procedure in Saveri et al. [195]. Formula trees are grown recursively by sampling production rules with fixed probabilities, which allows the grammar to introduce substantial variability while still enforcing constraints on maximum depth, number of nodes, and number of variables appearing in predicates. These constraints prevent the generation of overly complex or unintelligible formulae, and ensure that the resulting temporal patterns remain interpretable. The grammar itself may produce a large and diverse pool of candidate concepts, spanning a wide range of temporal behaviours.
Algorithm 1, in Appendix A, reports the grammar-based template generation procedure. Starting from a base set of atomic propositions, increasingly complex STL formula templates are constructed by recursively applying unary temporal and logical operators (e.g. $\mathbf{F}$, $\mathbf{G}$, $\neg$) and binary operators (e.g. $\wedge$, $\vee$, $\mathbf{U}$). The recursion is bounded by a maximum number of syntactic nodes and by a constraint on the number of signal dimensions that may appear in each formula. At each expansion step, production rules are sampled according to fixed probabilities, yielding a stochastic yet controlled exploration of the formula space.

Each generated template is subsequently instantiated over a grid of parameter values, producing a large pool of concrete STL formulae. Since this pool may contain many behaviourally redundant concepts, a selection step is applied based on the similarity of robustness signatures computed over a representative set of trajectories. Only formulae that are sufficiently dissimilar from those already selected are retained, resulting in a compact, semantically diverse, and interpretable concept set. Full details of the template generation and selection procedure are provided in Appendix A.

However, grammar-based sampling also tends to introduce redundancy, as many syntactically distinct formulae can behave very similarly when evaluated over the same dataset. To address this, every generated formula is evaluated on a reference set of trajectories $\mathcal{T}$ to obtain a robustness profile,

$$r(\varphi) = (\rho(\varphi, \tau_1), \ldots, \rho(\varphi, \tau_m)), \quad \tau_i \in \mathcal{T}$$

which effectively summarises the behaviour of the formula over the dataset. Formulae with highly correlated robustness profiles are considered behaviourally redundant. In practice, two formulae φ and ψ are treated as duplicates when the cosine similarity between their robustness vectors exceeds a predefined threshold, $\cos(r(\varphi), r(\psi)) \geq 1 - \delta$, and only one of them is kept. This step ensures that the final concept set achieves behavioural diversity rather than merely syntactic variation.

Redundancy filtering relies on both behavioural similarity and structural simplicity. When multiple formulae exhibit nearly identical robustness profiles, indicated by cosine similarity exceeding $1-\delta$, only the structurally simplest formula is retained, as measured by node count. This ensures that the concept set remains compact whilst preserving behavioural diversity: syntactically complex formulae that behave identically to simpler alternatives do not contribute additional interpretability and are therefore discarded. The result is a concept set where each formula represents a distinct temporal behaviour and is expressed in its most readable form.

To connect this construction to the running example, the formulae $\varphi_1 = \mathbf{F}_{[15,25]}(x >$

2) and $\varphi_2 = \mathbf{G}_{[0,50]}(x < 1.5)$ can be viewed as two candidate concepts sampled from the grammar and retained after redundancy filtering. Although simple, they illustrate how distinct temporal structures, such as local threshold exceedance versus global amplitude constraint, correspond to structurally different formula trees. In practice, the grammar generates a much richer and more diverse concept set, but the same construction principles apply.

Finally, a practical consideration concerns differences in dataset length and sampling rate. The grammar samples temporal interval endpoints from a fixed normalised range $[0, 100]$, independent of any specific dataset. When applying these formulae to trajectories of different lengths, the interval endpoints are rescaled proportionally to the signal duration. For instance, a formula with interval $[10, 30]$ applied to a trajectory of length 200 time steps becomes $[20, 60]$ after rescaling. This approach ensures that temporal operators maintain consistent semantic interpretation across datasets with different temporal resolutions, allowing the same concept set to generalise across varying signal lengths without requiring dataset-specific grammar modifications.

Importantly, this design avoids introducing implicit assumptions about absolute time scales or fixed sampling frequencies. Temporal properties are defined in relative, rather than dataset-dependent, units, so that a constraint expressing, for example, an early, mid, or late behavioural transition retains its meaning irrespective of the concrete sequence length. As a result, STELLE does not assume homogeneous signal duration across samples, nor does it rely on fixed temporal horizons. The only requirement is that trajectories admit a consistent discrete time index, enabling robustness evaluation after proportional rescaling. This scale-consistent formulation supports cross-dataset generalisation while preserving the semantic integrity of the temporal concepts.

8.3 Trajectory Embedding Kernel

A central component of the proposed framework is the construction of a shared semantic space in which trajectories and STL formulae can be compared directly. This is made possible through a trajectory embedding kernel inspired by an existing kernel for STL formulae proposed in the literature. In this chapter, we first revisit the STL kernel introduced in prior work, which provides a functional-analytic foundation for embedding logical formulae into a Hilbert space based on their robustness semantics. We then introduce the trajectory embedding kernel that extends this concept to raw time series data, enabling joint reasoning over trajectories and temporal logic concepts within a unified and interpretable space.

Motivation

The goal of STELLE is to build a concept-based classifier whose internal representations are grounded in STL semantics. This requires representing both trajectories and formulae in a common embedding space where distances and similarities reflect behavioural alignment with respect to temporal properties.

The original STL kernel [196] provides a mechanism for embedding STL formulae into

a continuous space using robustness values. Each formula φ is mapped to the functional

$$\rho(\varphi, \cdot) : \mathcal{T} \to \mathbb{R},$$

which assigns to every trajectory $\tau \in \mathcal{T}$ the robustness $\rho(\varphi, \tau)$. To define inner products between such functionals, it is necessary to introduce a probability measure μ_0 over the trajectory space $\mathcal{T}$, which specifies how agreement between behaviours is aggregated across trajectories. This viewpoint allows STL formulae to be compared via an inner product in $L^2(\mathcal{T}, \mu_0)$.

For concept-based learning, however, it is equally important to embed trajectories into the same semantic space. The key idea is therefore to treat trajectories as functionals as well, defining for each reference trajectory τ a smooth similarity functional ρ_τ that plays a role analogous to robustness. This parallel construction allows us to define a kernel between any trajectory and any STL formula through their behavioural interaction with the entire trajectory space under μ_0.

Thus, the trajectory–formula kernel extends the original STL kernel and enables a unified representation of both symbolic and numerical objects, preserving interpretability without requiring black-box learned embeddings.

8.3.1 A Kernel for STL Formulae

We recall the definition of the STL kernel introduced in Bortolussi et al. [196]. Robustness allows each STL formula φ to be interpreted as a functional

$$\rho(\varphi, \cdot) : \mathcal{T} \to \mathbb{R}, \qquad \tau \mapsto \rho(\varphi, \tau),$$

mapping trajectories to real-valued robustness scores. The similarity between formulae can then be captured through an inner product structure analogous to the scalar product of finite-dimensional vectors. Just as the scalar product between vectors is large when they "agree" on many components, a functional inner product can be defined to measure agreement over the space of trajectories. However, unlike finite-dimensional vectors where each component is weighted equally, integrating over uncountably many trajectories requires imposing a probability measure μ_0 on the trajectory space $\mathcal{T}$. This measure serves two purposes: it makes the integral well-defined, and it encodes a preference over which trajectories are deemed important for comparison. The original authors define μ_0 to favour "simple" trajectories, using total variation as a measure of simplicity. This ensures that the kernel emphasises behavioural similarity over smooth, representative trajectories rather than pathological or overly complex signals. Given this measure, the similarity between two formulae φ and ψ is defined as:

$$k(\varphi, \psi) = \langle \rho(\varphi, \cdot), \rho(\psi, \cdot) \rangle = \int_{\tau \in \mathcal{T}} \rho(\varphi, \tau)\, \rho(\psi, \tau)\, d\mu_0(\tau). \tag{8.1}$$

This construction embeds STL formulae into the Hilbert space $L^2(\mathcal{T}, \mu_0)$, where inner products quantify behavioural similarity: the kernel is large when φ and ψ behave similarly over high-probability trajectories, and negative when they systematically disagree.

It is important to stress that μ_0 does not encode any dataset-specific distributional assumptions, nor does it incorporate class labels or empirical statistics from the training

set. The measure serves a purely functional-analytic role: it renders the integral well-defined and induces a notion of behavioural similarity over the space of admissible trajectories. Although μ_0 favours trajectories with bounded total variation, this bias reflects a preference for structurally simple signals in the abstract trajectory space $\mathcal{T}$, rather than an assumption about the generative process underlying the observed data. Consequently, the kernel construction does not presuppose stationarity, Gaussianity, or any parametric form of the data distribution. It defines similarity in semantic terms via robustness agreement, rather than in statistical terms tied to a specific dataset.

This kernel provides the theoretical foundation for STELLE by establishing a continuous, semantics-preserving embedding of logical formulae. The trajectory embedding kernel derived next builds on this functional viewpoint.

Kernel transformations for concept selection. In practice, the kernel values computed via Equation 8.1 are subject to two transformations before being used for concept selection. First, the kernel is *normalised* to obtain

$$\bar{k}(\varphi, \psi) = \frac{k(\varphi, \psi)}{\sqrt{k(\varphi, \varphi) \cdot k(\psi, \psi)}},$$

which ensures that similarity values lie in a bounded range and removes scale effects introduced by formulae with large robustness magnitudes. Second, the normalised kernel is *exponentiated* using a Gaussian transformation:

$$\tilde{k}(\varphi, \psi) = \exp\left(-\frac{d^2_{\mathcal{H}}(\varphi, \psi)}{2\sigma^2}\right),$$

where $d^2_{\mathcal{H}}(\varphi, \psi) = k(\varphi, \varphi) + k(\psi, \psi) - 2k(\varphi, \psi)$ is the squared distance in the reproducing kernel Hilbert space (RKHS), and $\sigma^2 > 0$ controls the bandwidth of the transformation. This exponential mapping amplifies differences between moderately similar formulae, enhancing contrast in the similarity structure and promoting behavioural diversity during redundancy filtering in concept selection. Normalisation stabilises the selection process by reducing sensitivity to outliers, whilst exponentiation selectively sharpens distinctions amongst formulae with intermediate similarity. The combined effect produces concept sets with richer temporal diversity, as demonstrated empirically in Chapter 10, where we analyse the impact of these transformations on classification performance.

It is important to note that these kernel transformations are applied exclusively within the concept selection and post-processing stages of the framework. The STL kernel is not used during model training, nor does it influence gradient-based optimisation or classifier parameter updates. As a result, normalisation and exponentiation affect only the structure of the symbolic concept space, specifically how redundancy and diversity among STL formulae are assessed, without impacting the learning dynamics or decision boundaries of the classifier.

8.3.2 Trajectory Embedding Kernel

While the STL kernel embeds logical formulae into $L^2(\mathcal{T}, \mu_0)$, STELLE requires that raw trajectories also inhabit this same semantic space. This unified embedding is

essential for concept-based learning: it allows the model to compare trajectories directly with STL formulae, measure their behavioural alignment, and construct interpretable representations where both symbolic specifications and numerical data are treated as first-class objects. Without such an embedding, trajectories and formulae would remain in fundamentally different mathematical spaces, preventing the direct interaction necessary for logically grounded classification.

To achieve this unification, a trajectory-dependent functional ρ_τ is defined for every trajectory $\tau \in \mathcal{T}$, playing a role analogous to the robustness functional $\rho(\varphi, \cdot)$ for STL formulae. This construction enables the definition of a trajectory–formula kernel that extends the original STL kernel whilst preserving its theoretical properties.

Similarity functional for trajectories

For a fixed reference trajectory $\tau \in \mathcal{T}$, the similarity functional $\rho_\tau : \mathcal{T} \to \mathbb{R}$ is defined through a normalised distance measure. Let

$$d_\tau(\xi) = \|\xi - \tau\|_2^2$$

denote the squared L^2 distance between trajectories ξ and τ. The similarity functional is then constructed as

$$\rho_\tau(\xi) = \frac{d_\tau(\xi)}{\varepsilon},$$

where $\varepsilon > 0$ is a scaling parameter that controls the behaviour of the functional across three dimensions:

- *Locality.* Smaller ε causes $\rho_\tau(\xi)$ to decay more sharply with distance, emphasising nearby trajectories;
- *Smoothness.* Larger ε produces smoother similarity profiles, reducing sensitivity to small perturbations;
- *Numerical stability.* Appropriate choice of ε prevents degenerate gradients when $\xi \approx \tau$ and avoids overflow for distant trajectories.

This definition mirrors the interpretation of STL robustness: just as $\rho(\varphi, \xi)$ quantifies how strongly trajectory ξ satisfies temporal property φ, the quantity $\rho_\tau(\xi)$ quantifies how close ξ is to the reference trajectory τ. Both ρ_τ and $\rho(\varphi, \cdot)$ are thereby treated as elements of the same functional space $L^2(\mathcal{T}, \mu_0)$.

Normalisation of robustness values. Because $d_\tau(\xi)$ may vary significantly across trajectories or datasets, the similarity functional ρ_τ is normalised before computing kernel values. Following the same approach as for STL formulae (Section 8.3.1), we apply

$$\bar{\rho}_\tau(\xi) = \frac{\rho_\tau(\xi)}{\sqrt{\rho_\tau(\tau) \cdot \|\rho(\Phi_C, \xi)\|_2}},$$

where $\|\rho(\Phi_C, \xi)\|_2$ represents the norm of the concept activation vector for trajectory ξ. This normalisation prevents scale imbalances when comparing trajectory similarity

$\rho_\tau(\xi)$ to formula robustness $\rho(\varphi, \xi)$ within the shared embedding space, ensuring that both types of functionals contribute proportionally to kernel computations.

Importantly, unlike the formula kernel, we do *not* apply exponential transformation to ρ_τ values. Whilst exponentiation enhances contrast amongst formulae during concept selection (Section 8.3.1), exponentiating trajectory similarities introduces unnecessary amplification of small differences between trajectories, potentially creating a noisier embedding space without corresponding gains in discriminative power. The ablation studies in Chapter 10 demonstrate that normalisation alone produces stable, well-calibrated trajectory embeddings, and that additional exponential transformations degrade classification performance by over-emphasising local variations that do not reflect meaningful temporal distinctions.

Selection of ε. The scaling parameter ε plays a crucial role in determining the effective resolution of the embedding. If ε is too small, the kernel becomes overly sensitive to minor differences between trajectories, leading to overfitting and poor generalisation. Conversely, if ε is too large, the kernel fails to distinguish between genuinely different behavioural patterns.
In this work, ε is initialised to 1 and subsequently treated as a learnable parameter. Its value is optimised jointly with the remaining model parameters through an explicit term in the training objective, allowing the model to adapt the effective scale of the embedding to the data. This formulation avoids manual tuning of ε and provides a principled mechanism for balancing sensitivity to behavioural differences with robustness to noise.

Trajectory–Formula Kernel

Once both trajectories and formulae are associated with functionals in $L^2(\mathcal{T}, \mu_0)$, the inner product structure naturally extends to define a trajectory–formula kernel. For a trajectory τ and an STL formula φ, the kernel is given by

$$k(\tau, \varphi) = \int_{\xi \in \mathcal{T}} \rho_\tau(\xi)\, \rho(\varphi, \xi)\, \mathrm{d}\mu_0(\xi). \tag{8.2}$$

This definition inherits the functional-analytic properties of the STL kernel and provides a principled measure of behavioural alignment between trajectories and temporal specifications.

The trajectory–formula kernel satisfies three essential properties:

1. *Shared semantic space.* Both trajectories and STL formulae are embedded into the same Hilbert space $L^2(\mathcal{T}, \mu_0)$, enabling direct comparison and concept-based reasoning. This unification is fundamental to STELLE's architecture: it allows the model to measure how strongly a trajectory exhibits a particular temporal property, expressed as a kernel value rather than an abstract latent representation.

2. *Behavioural grounding.* The kernel $k(\tau, \varphi)$ measures similarity not through superficial features but through the way both τ and φ interact with the entire trajectory space under μ_0. Two objects are close if they behave similarly across $\mathcal{T}$: a trajectory τ has high kernel value with a formula φ if trajectories similar to τ tend to satisfy φ strongly. This captures a notion of behavioural alignment that respects the semantics of temporal logic.

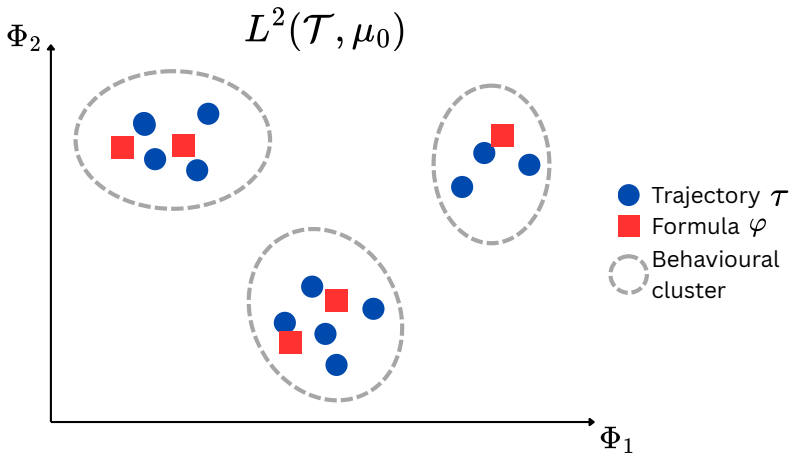


Figure 8.2: Visualisation of the shared embedding space $L^2(\mathcal{T}, \mu_0)$ where both trajectories (blue circles) and STL formulae (red squares) are embedded as functionals. Objects cluster according to behavioural similarity: trajectories and formulae that interact similarly with the trajectory space under μ_0 are positioned nearby. Dashed lines indicate high kernel values $k(\tau, \varphi)$, representing strong behavioural alignment. The axes represent projections onto basis formulae Φ_1 and Φ_2, illustrating how the space is interpretable through temporal logic concepts.

3. *Inherited theoretical guarantees.* As a direct extension of the original STL kernel, the trajectory–formula kernel inherits positive definiteness and the structural properties of the L^2 inner product. This ensures that distances and similarities computed in the embedded space are well-defined, stable, and consistent with the functional-analytic framework.

Figure 8.2 illustrates this shared semantic space, where trajectories and formulae cluster according to their behavioural profiles over $\mathcal{T}$. Trajectories that exhibit similar temporal patterns are positioned near the formulae that characterise those patterns, providing an intuitive geometric interpretation of the concept-based representation.

Running example. To relate this construction to the running example introduced earlier, consider the trajectory τ and the STL formulae

$$\varphi_1 = \mathbf{F}_{[15,25]}(x > 2) \qquad \mathbf{G}_{[0,50]}(x < 1.5).$$

The kernel value $k(\tau, \varphi_1)$ is large because trajectories that are behaviourally close to τ under μ_0 also tend to exhibit a threshold exceedance within the interval $[15, 25]$. By contrast, $k(\tau, \varphi_2)$ is small (or negative), reflecting the systematic violation of the global amplitude constraint by trajectories similar to τ. The trajectory–formula kernel therefore measures behavioural alignment across the trajectory space rather than mere pointwise satisfaction on a single signal.

Computational realisation. Whilst the trajectory–formula kernel is defined through the integral in Equation 8.2, STELLE does not explicitly evaluate this integral during training or inference. Instead, the kernel structure is realised through Monte Carlo

approximation: a finite set of N sample trajectories $\{\xi_1, \ldots, \xi_N\}$ is drawn from μ_0, and the kernel is approximated as

$$k(\tau, \varphi) \approx \frac{1}{N} \sum_{i=1}^{N} \rho_\tau(\xi_i)\, \rho(\varphi, \xi_i).$$

This approximation converges to the true kernel value as $N \to \infty$ by the law of large numbers, and in practice $N = 5{,}000$ provides sufficient accuracy whilst remaining computationally tractable.

For a set of C STL formulae $\Phi = \{\varphi_1, \ldots, \varphi_C\}$ and a trajectory τ, the embedding of τ into the concept space is then given by the vector

$$k(\tau, \Phi) = \big(k(\tau, \varphi_1), \ldots, k(\tau, \varphi_C)\big)^\top \in \mathbb{R}^C, \tag{8.3}$$

which quantifies the behavioural alignment of τ with each temporal concept in Φ. This embedding forms a key element to the STELLE framework, described in Chapter 8.4.

8.3.3 Comparison with Alternative Embeddings

The trajectory embedding kernel differs fundamentally from existing approaches. Deep embeddings produced by neural encoders such as CNNs, RNNs, and Transformers are undeniably useful, yet they lack symbolic structure, do not align with formal semantics, and require post-hoc explanations to make their representations interpretable [7, 50, 77]. Shapelet methods, while effective at identifying discriminative subsequences in time series, cannot express temporal relationships such as ordering, duration, nesting, or logical combinations of patterns [66, 197]. Similarly, motif-based distances capture repeated patterns in sequences but do not provide semantic grounding or support robustness-based reasoning about temporal behaviour [198, 199]. The trajectory–formula kernel is unique in that it embeds trajectories and logical formulae into the same space, with similarity defined through behavioural profiles over $\mathcal{T}$. This design preserves interpretability by construction, enabling direct semantic comparison between trajectories and specifications while maintaining the formal guarantees of temporal logic. Unlike black-box embeddings, the kernel provides transparent reasoning about temporal patterns, and unlike pattern-matching methods, it supports compositional semantics that can express complex temporal relationships.

8.4 The STELLE Architecture

This chapter presents the full architecture of STELLE (Signal Temporal Logic Embedding for Logically-grounded Learning and Explanation). The model performs time series classification through concept-based reasoning grounded in the semantics of Signal Temporal Logic (STL). Its design maintains interpretability at every stage: trajectories are embedded into a concept space defined by STL formulae, discriminability is assessed through robustness-based atypicality, and concept relevance is learned explicitly. The final classifier operates on these interpretable representations, ensuring that explanations follow directly from the model's internal computations.

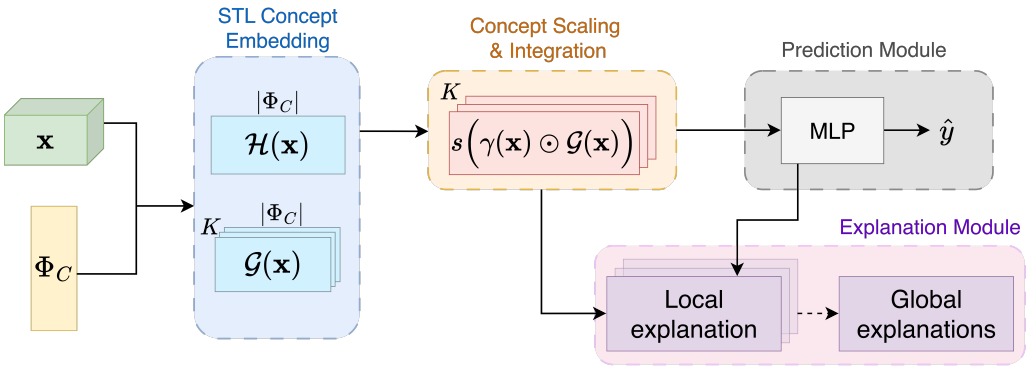


Figure 8.3: STELLE architecture. Input trajectories $\mathbf{x}$ are embedded via STL concepts formulae Φ, creating $\mathcal{H}(\mathbf{x})$. This gets scaled, creating $\gamma(\mathbf{x})$, and integrated with class-specific scores $\mathcal{G}(\mathbf{x})$ for classification $\hat{y}$ and generation of local explanations, which can be aggregated into global explanations.

8.4.1 Overall Architecture

The STELLE pipeline consists of four main components, as depicted in Figure 8.3: the trajectory embedding, the discriminability mechanism, the learned concept relevance module, and the classifier head. Given an input trajectory $\mathbf{x}$, the system first computes its robustness activations with respect to a predefined STL concept set, constructed following the procedure described in Chapter 8.2. These activations form a concept-level representation of the trajectory, which is then modulated through a concept relevance layer that selects or attenuates concepts. A discriminability mechanism further highlights atypical concept activations that are informative for classification. The resulting representation is passed to a lightweight multilayer perceptron (MLP) that outputs class scores.

The architecture is intentionally modular, with each component contributing to interpretability: robustness-based embeddings ensure symbolic grounding, the discriminability mechanism emphasises deviations in concept behaviour, the relevance layer exposes which concepts the model considers important, and the classifier head remains shallow enough for its influence to be traceable.

8.4.2 Trajectory Embedding Layer

The first stage of the architecture transforms the raw trajectory into concept activations. Let $\Phi = \{\varphi_1, \ldots, \varphi_C\}$ denote the fixed set of STL concept formulae. For an input trajectory $\mathbf{x}$, the trajectory embedding (introduced in Chapter 8.3.2) is defined as

$$\mathcal{H}(\mathbf{x}, \Phi) = \mathcal{H}(\mathbf{x}) = \big(k(\mathbf{x}, \varphi_1), \ldots, k(\mathbf{x}, \varphi_C)\big)^\top \in \mathbb{R}^C, \tag{8.4}$$

as per Equation 8.3, which maps $\mathbf{x}$ into the shared semantic space induced by the trajectory–formula kernel. Each coordinate is the kernel similarity of the trajectory into a human-interpretable temporal formula. This yields a symbolic, interpretable representation where every dimension corresponds to a well-defined STL concept.

8.4.3 Discriminability Mechanism

While the trajectory embedding captures how strongly each temporal concept is satisfied, not all concepts are equally informative for distinguishing between classes. The discriminability mechanism quantifies, in a principled and statistically grounded way, how atypical a concept's robustness value is for each class, thereby turning raw robustness into class-specific evidence.

Let $\mathcal{T}_k \subset \mathcal{T}$ denote the set of training trajectories belonging to class k, and define

$$\mu_{k,i} = \mathbb{E}_{\mathbf{x}\sim\mathcal{T}_k}[\rho(\varphi_i, \mathbf{x})]\,, \qquad \sigma_{k,i} = \mathrm{std}_{\mathbf{x}\sim\mathcal{T}_k}[\rho(\varphi_i, \mathbf{x})]\,,$$

as the empirical mean and standard deviation of robustness values for concept φ_i within class k. For a new trajectory $\mathbf{x}$, the discriminability of φ_i with respect to class k is defined as the normalised deviation from the robustness statistics of the *other* classes:

$$\mathcal{G}_i^k(\mathbf{x}) = \frac{\left|\rho(\varphi_i, \mathbf{x}) - \mu_{k^*,i}\right|}{\sigma_{k^*,i} + \varepsilon_G}, \qquad k^* \neq k, \tag{8.5}$$

where $\varepsilon_G > 0$ is a very small constant (typically 10^{-8}) that ensures numerical stability by preventing division by zero when $\sigma_{k^*,i}$ is negligible.

Interpretation. Large values of $\mathcal{G}_i^k(\mathbf{x})$ indicate that the observed robustness $\rho(\varphi_i, \mathbf{x})$ is unusually high or unusually low compared to what is typical for class k, providing evidence *against* that class. Conversely, small values signal that the concept behaves as expected for class k, providing supportive evidence.

This mechanism is closely related to statistical hypothesis testing: it assesses the atypicality of a concept relative to class-conditional robustness distributions. However, unlike classical tests, the mechanism is fully differentiable, class-aware, and performed directly at the level of temporal logic concepts rather than raw features. As a result, STELLE attributes class evidence to interpretable temporal properties, ensuring that discriminability remains transparent, faithful, and semantically grounded.

Running example. To clarify the role of discriminability, consider again the running example trajectory τ and the concept $\varphi_1 = \mathbf{F}_{[15,25]}(\mathbf{x} > 2)$. Suppose that, over the training data, the empirical robustness statistics for φ_1 are

$$\mu_{0,1} = 0.7, \quad \sigma_{0,1} = 0.1 \qquad \text{for class } 0,$$

and

$$\mu_{1,1} = -0.6, \quad \sigma_{1,1} = 0.2 \qquad \text{for class } 1.$$

If the example trajectory satisfies $\rho(\varphi_1, \tau) = 0.8$, its robustness lies close to the typical value observed for class 0 and far from that of class 1. The discriminability mechanism therefore assigns a high score to φ_1 for class 0 and a low (or negative) score for class 1.

In contrast, consider $\varphi_2 = \mathbf{G}_{[0,50]}(\mathbf{x} < 1.5)$, which the trajectory violates with negative robustness. If both classes exhibit similarly negative robustness values for φ_2, its deviation from class means will be small for all classes, and the resulting discriminability score will be low. In this case, the concept is activated but not class-informative.

This example highlights the conceptual distinction between *activation* and *discriminative value*. Robustness measures whether a temporal property holds for a specific trajectory; discriminability measures whether that property meaningfully separates classes. Only concepts that are both activated and statistically distinctive contribute strong class-specific evidence.

8.4.4 Learned Concept Relevance

While discriminability quantifies how strongly each concept supports or contradicts a class, not all concepts are equally important for every trajectory. STELLE therefore learns a relevance vector that adaptively modulates the contribution of each temporal concept to the final decision.

Given the trajectory embedding $\mathcal{H}(\mathbf{x})$ defined in Equation 8.4,
the model computes a temperature-scaled relevance score

$$\gamma(\mathbf{x}) = \frac{\mathcal{H}(\mathbf{x})}{T} \in \mathbb{R}^C, \qquad T > 0, \tag{8.6}$$

where the temperature parameter T controls the scale of the relevance modulation. Dividing $\mathcal{H}(\mathbf{x})$ by T uniformly rescales all concept activations, amplifying or attenuating their relative contrast without inducing sparsity. Smaller values of T increase the separation between strongly and weakly activated concepts, while larger values produce a more uniform modulation. In practice, T ensures scale compatibility between trajectory-dependent activations and downstream discriminability scores, contributing to numerical stability during training.

Modulation of class-specific evidence. The relevance vector interacts multiplicatively with the discriminability matrix through a Hadamard product:

$$\mathbf{z}(\mathbf{x}) = \gamma(\mathbf{x}) \odot \mathcal{G}(\mathbf{x}) \in [0, \infty)^{C \times K},$$

thereby amplifying discriminative concepts and suppressing irrelevant ones. Each entry $\mathbf{z}_i^k(\mathbf{x})$ combines:

- how strongly φ_i activates on $\mathbf{x}$ (via γ), and
- how atypically it behaves for class k (via $\mathcal{G}$),

yielding a concept-level score of class-specific evidential strength.

To produce a bounded, signed representation that reflects both positive and negative support, the model applies a Softsign activation:

$$\bar{\mathbf{z}}(\mathbf{x}) = s(\mathbf{z}(\mathbf{x})) \in (-1, 1)^{C \times K}, \qquad s(u) = \frac{u}{1 + |u|}.$$

The resulting matrix $\bar{\mathbf{z}}(\mathbf{x})$ is finally flattened and passed to the classifier head, completing a pipeline in which concept activation, discriminability, and relevance interact in a fully interpretable and logically grounded manner.

Why Softsign? Softsign is selected for three key reasons:

1. *Symmetry.* It preserves both positive and negative evidence, a crucial requirement for concept-level interpretability.
2. *Smoothness.* It is differentiable everywhere, ensuring stable optimisation.
3. *Controlled compression.* Large values are gently compressed, preventing dominance by extreme discriminability scores.

Alternative nonlinearities would break these interpretability guarantees:

- ReLU would zero-out all negative evidence, preventing the model from expressing that a concept contradicts a class.
- Sigmoid would map everything to $(0, 1)$, eliminating the distinction between supportive and opposing concepts and collapsing interpretive structure.

Softsign therefore provides the right balance: it preserves the polarity of concept evidence, stabilises the scale of concept interactions, and maintains the mathematical semantics of the relevance–discriminability combination.

8.4.5 Classifier Head

After relevance modulation and Softsign normalisation, the matrix $\bar{\mathbf{z}}(\mathbf{x}) \in (-1, 1)^{C\times K}$ encodes the concept-level evidential structure for all classes. For classification, this matrix is flattened into a single vector:

$$\bar{\mathbf{z}}_{\text{flat}}(\mathbf{x}) \in (-1, 1)^{CK},$$

which serves as the input to a compact MLP.

The classifier head is deliberately kept shallow, typically one or two hidden layers of modest width, to preserve a transparent relationship between concept-level evidence and final logits. Instead, the architecture relies on the expressiveness of the STL concept space and the discriminability–relevance mechanism. The MLP's role is purely to integrate these signals in a stable and controlled manner.

Let $F : \mathbb{R}^{CK} \to [0, 1]^K$ denote the classifier. The final logit vector is:

$$\ell(\mathbf{x}) = F\left(\bar{\mathbf{z}}_{\text{flat}}(\mathbf{x})\right),$$

and the predicted class is obtained via the softmax operator:

$$\hat{y} = \arg\max_k \ell_k(\mathbf{x}).$$

Regularisation. Standard techniques such as weight decay or dropout may be used inside the MLP without interfering with the interpretability of concept relevance, as these regularisation methods operate solely on the classifier head and do not affect the semantics of the concept-level representation $\bar{\mathbf{z}}(\mathbf{x})$. In practice, mild L_2 regularisation is preferred for its simplicity and stability.

Overall, the classifier head acts as a lightweight aggregator: it consolidates concept-level evidence while ensuring that the final prediction remains directly traceable to interpretable STL components.

8.4.6 Interpretability by Design

Interpretability is woven into the architecture rather than imposed through post-hoc analyses. Each component contributes to this goal:

- *Robustness-based embeddings* ensure that the model operates in a semantic space defined by formal temporal concepts.
- *The discriminability mechanism* highlights atypical temporal behaviours that support or contradict each class, providing intuitive diagnostic evidence.
- *Learned relevance scores* expose which concepts are important for the model, enabling transparent inspection of per-class and per-instance behaviour.
- *The shallow classifier head* preserves traceability between concept activations and predictions.

This stands in marked contrast to post-hoc interpretability methods, which attempt to explain decisions after training by approximating the influence of features or perturbing inputs. Such approaches may fail to reflect the true internal logic of the model. In STELLE, explanations are derived directly from the model's computation, ensuring faithfulness, transparency, and stability. Because each component remains grounded in the semantics of STL, the resulting explanations take the form of human-interpretable temporal statements rather than abstract vectors or saliency maps.

Overall, the architecture realises the objective of concept-based interpretability without sacrificing classification performance, demonstrating that logical expressiveness and modern learning techniques can be integrated into a unified, semantically meaningful system.

8.5 Explanation Framework

The STELLE architecture supports interpretable time series classification by expressing explanations in terms of human-readable temporal logic concepts. Explanations in STELLE arise naturally from the model's internal symbolic operations: concept activations derived from robustness, discriminability scores quantifying concept atypicality per class, and integrated gradients tracing the influence of each concept on the final logits. This chapter describes how these components are combined to extract faithful *local* explanations for individual trajectories and *global* explanations that characterise entire classes.

8.5.1 Local Explanations

Local explanations describe why the model assigns a specific class to an input trajectory. In STELLE, a prediction is produced by combining three distinct factors: (i) which temporal concepts are relevant for the specific trajectory, (ii) how discriminative these concepts are with respect to each class, and (iii) how the classifier combines this information to produce its final decision.

Accordingly, local explanations are constructed by explicitly modelling and integrating these three effects.

Specifically, STELLE relies on:

1. *Concept relevance* $\gamma(\mathbf{x})$, which identifies which temporal concepts are activated by the trajectory $\mathbf{x}$;
2. *Class discriminability* $\mathcal{G}(\mathbf{x})$, which measures how atypical each activated concept is for each class;
3. *Classifier sensitivity*, which captures how variations in concept activations affect the classifier's output.

The first two quantities are intrinsic to the symbolic embedding. The third must be recovered from the classifier itself: we therefore estimate it using gradient-based attribution, allowing us to analyse the model's behaviour directly in the concept space rather than in raw signal space.

These components interact within the latent concept space

$$\mathbf{z}(\mathbf{x}) = \gamma(\mathbf{x}) \odot \mathcal{G}(\mathbf{x}) \in [0, \infty)^{C \times K},$$

where $\mathbf{z}_{i,k}(\mathbf{x})$ quantifies how strongly concept φ_i supports class k for trajectory $\mathbf{x}$. This representation preserves the original semantics of robustness while exposing how evidence is accumulated across concepts and classes. To extract explanations, this latent representation is further weighted by the gradient-based attributions, yielding a concept-class activation matrix that integrates activation strength, discriminability, and gradient-based importance for the final classification.

Attribution through Integrated Gradients

To quantify classifier sensitivity at the level of temporal concepts, we apply Integrated Gradients (IG) (introduced in Section 4.3.2) to the latent representation $\mathbf{z}(\mathbf{x})$.

Although several attribution methods can in principle be used in our setting, we adopt Integrated Gradients for two main reasons. First, IG satisfies desirable axioms such as sensitivity and implementation invariance, which ensure that concept attributions are stable and reflect genuine functional dependencies rather than artefacts of parametrisation. Secondly, IG proves particularly robust in concept-based models, where gradient saturation or bounded activations may distort relevance estimates: the integration path mitigates such issues by capturing contributions across multiple scales of activation.

Let the classifier head be a function

$$F : \mathbb{R}^{CK} \to [0, 1]^K,$$

mapping the flattened latent matrix to class probabilities, and let F_k denote the logit or probability associated with class k.

For an input $\mathbf{x}$ with latent representation $\mathbf{z}(\mathbf{x})$, we define a baseline $\mathbf{z}'$ (set to the zero vector) and compute, for each concept index i,

$$W_{k,i} = (z_i(\mathbf{x}) - z'_i) \int_0^1 \frac{\partial F_k(\mathbf{z}' + \alpha(\mathbf{z}(\mathbf{x}) - \mathbf{z}'))}{\partial z_i} \, d\alpha.$$

This yields a smooth and faithful estimate of the influence of concept i on F_k.

A key design choice is that the gradients are applied to the *unbounded* representation $\mathbf{z}(\mathbf{x})$, not to the bounded Softsign-transformed $\bar{\mathbf{z}}(\mathbf{x})$. This choice prevents saturation: the Softsign function compresses large values, which would otherwise mask differences in magnitude that are meaningful for attribution. By computing IG on $\mathbf{z}(\mathbf{x})$, explanations remain faithful to the quantitative relationships encoded by robustness and discriminability.

The resulting attribution weight $W_{k,i}$ measures how changes in the activation of concept φ_i would affect the confidence assigned to class k.

A Discriminative Scoring Mechanism

While IG quantifies how each concept influences the model logits, local explanations must also capture how distinctive each concept is for the target class. To this end, we compute a concept–class activation matrix by combining raw concept activations with their class-specific relevance weights:

$$\mathbf{A}(\mathbf{x}) = \mathbf{z}(\mathbf{x}) \odot \mathbf{W} \in [0, \infty)^{C \times K},$$

where $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^C$ contains the non-normalised concept activations, and $\mathbf{W} \in \mathbb{R}^{K \times C}$ is the matrix of IG attributions. Thus, $\mathbf{A}_{i,k}(\mathbf{x})$ reflects the contribution of concept φ_i to classifying $\mathbf{x}$ as class k, after integrating both activation strength and relevance.

To identify the concepts that best distinguish class k from the alternatives, we compute a discriminative score

$$\mathbf{r}(\mathbf{x})_k = \left| \mathbf{A}_{:,k}(\mathbf{x}) - \frac{1}{K-1} \sum_{k^* \neq k} \mathbf{A}_{:,k^*}(\mathbf{x}) \right|. \tag{8.7}$$

A concept receives a large score when its class-k activation differs substantially from the mean activation across all other classes. In other words, $\mathbf{r}(\mathbf{x})_k$ measures *distinctiveness*, complementing IG, which measures *influence*.

Alternative aggregation strategies, such as subtracting the maximum activation (rather than the mean) or using the raw activation $\mathbf{A}_{:,k}(\mathbf{x})$ directly without comparison, are explored in Section 10.6.1.

This combination ensures that selected concepts are not only impactful for the model's decision, but also genuinely characteristic of the target class.

Running example. Returning to the running example introduced earlier, consider the trajectory τ and the concepts

$$\varphi_1 = \mathbf{F}_{[15,25]}(x > 2) \qquad \varphi_2 = \mathbf{G}_{[0,50]}(x < 1.5).$$

Suppose the model predicts class 0 for τ. The concept φ_1 is strongly activated (positive robustness), statistically characteristic of class 0, and receives a high attribution weight $W_{0,1}$ from Integrated Gradients. Conversely, φ_2 is weakly discriminative and receives negligible attribution.

As a result, the combined activation

$$\mathbf{A}_{1,0}(\tau) = \mathbf{z}_{1,0}(\tau)\, W_{0,1}$$

is large, whereas $\mathbf{A}_{2,0}(\tau)$ remains small. The explanation mechanism therefore identifies φ_1 as the dominant temporal property supporting the prediction.

Selecting the most Influential Concepts

To ensure that local explanations remain concise, readable, and focused on the most informative temporal patterns, only a subset of the extracted concepts is retained. STELLE supports two complementary selection strategies, both operating on the relevance scores $\mathbf{r}$:

- *Fixed-budget selection.* Given a budget $\bar{\gamma} \in \mathbb{N}$, retain the $\bar{\gamma}$ concepts with highest relevance scores $\mathbf{r}$.
- *Cumulative relevance selection.* Sort the relevance scores in descending order and take the smallest prefix whose cumulative sum reaches a target ratio $\bar{\gamma}_t \in (0, 1]$ of the total relevance mass.

The cumulative relevance threshold $\bar{\gamma}_t$ is applied during the initial concept extraction stage, where candidate STL formulae are ordered according to their relative importance and a preliminary subset is retained. At this stage, the selection is solely driven by ranking and does not yet impose any constraints on how well the retained concepts jointly characterise the target class. Additional mechanisms governing the construction and refinement of the final explanation are introduced later, during a dedicated post-processing step.
The effect of the parameter $\bar{\gamma}_t$ is analysed in Section 10.6.2.

The resulting set of selected concepts is denoted

$$\mathcal{F}_{\mathbf{x}} = \{\varphi_1, \ldots, \varphi_{|\mathcal{F}_{\mathbf{x}}|}\}.$$

Each element of $\mathcal{F}_{\mathbf{x}}$ corresponds to a temporal logic concept that is both influential for the model's decision on $\mathbf{x}$ and discriminative for the predicted class.

Each selected concept $\varphi_i \in \mathcal{F}_{\mathbf{x}}$ is then refined into a more concise and human-readable STL expression through the logic-aware post-processing procedure described in Section 8.5.3. This refinement step yields a simplified concept φ'_i and, accordingly, an updated concept set

$$\mathcal{F}'_{\mathbf{x}} = \{\varphi'_1, \ldots, \varphi'_{|\mathcal{F}'_{\mathbf{x}}|}\}.$$

After post-processing, the refined concepts are combined into a final local explanation expressed in conjunctive form:

$$E_\ell(\mathbf{x}, \hat{y}) = \bigwedge_{\varphi'_i \in \mathcal{F}'_{\mathbf{x}}} \varphi'_i,$$

where $\hat{y}$ denotes the predicted class. Each conjunct represents an individual temporal property that the model considers both relevant for the specific trajectory and indicative of the target class.

The resulting formula provides a *sufficient* explanation of the prediction: whenever all conditions in $E_\ell(\mathbf{x}, \hat{y})$ are satisfied, the model is likely to assign the same class $\hat{y}$. This construction therefore yields compact, faithful, and semantically grounded explanations

directly aligned with the model's internal reasoning. Finally, the resulting explanation formula is further refined through a data-aware simplification step, described in Section 8.5.3, which leverages empirical properties of the dataset to remove redundancies and improve readability while preserving semantic and discriminative consistency.

Importance parameter imp_ℓ. During the post-processing stage used to construct a local explanation from a set of candidate STL formulae, we introduce the importance parameter imp_ℓ to control the minimum level of separability required of the explanation. In particular, imp_ℓ specifies the fraction of data points that the extracted explanation is allowed to leave out of the classification. Setting $imp_\ell = 0$ enforces maximal fidelity, requiring the explanation to separate all relevant points, regardless of the resulting formula complexity. Higher values of imp_ℓ relax this requirement, allowing the explanation to cover only a subset of points and thereby favouring simpler formulae with fewer nodes. In this way, imp_ℓ provides an explicit and interpretable mechanism for trading off explanatory simplicity against exact class separation, by constraining the minimum proportion of points that must be correctly classified by the explanation. The effect of this parameter is analysed in Section 10.6.2.
Together, $\bar{\gamma}t$ and imp_ℓ regulate explanation complexity at complementary stages: the former controls relevance-based pruning during candidate extraction, while the latter acts as a stopping criterion during post-processing to ensure sufficient separability.

8.5.2 Global Explanations

While local explanations justify predictions for individual trajectories, global explanations describe the characteristic behaviours of entire classes. Global explanations do not depend on the model's predictions but are constructed using the true training labels, ensuring alignment with the dataset's semantics rather than with model-specific biases.

For class k, we collect all local explanations produced for trajectories with true label k:

$$\mathcal{F}_k = \{E_\ell(\mathbf{x}, k) \mid \mathbf{x} \in \mathcal{T}_{\text{train}},\ y(\mathbf{x}) = k\}.$$

Each explanation contributes a single STL formula, already expressed in conjunctive form, that was highly discriminative for that class at the instance level. We use the training set to construct global explanations because these trajectories were available during model training and thus their local explanations reflect patterns the model learned to associate with each class. The resulting global explanations are then validated on the test set by measuring their separation percentage, ensuring they generalise beyond the training distribution.

Division Analysis

The goal of division analysis is to identify which formulae from the local explanation pool remain discriminative at the global, class-level scale. A formula is considered discriminative for class k if its robustness values on class-k trajectories are systematically different from (i.e., fall outside the range of) robustness values observed on all other

classes. This ensures that the formula captures a temporal property that is characteristic of class k and not shared by other classes.

Formally, we construct a division matrix D^k indicating whether a formula behaves outside the robustness range of all non-class-k trajectories:

$$D_{ij}^k = \begin{cases} 1, & \text{if } \rho(\varphi_j, \mathbf{x}_i) \notin [\min_{\mathbf{x} \notin y^{-1}(k)} \rho(\varphi_j, \mathbf{x}),\ \max_{\mathbf{x} \notin y^{-1}(k)} \rho(\varphi_j, \mathbf{x})], \\ 0, & \text{otherwise.} \end{cases}$$

where $y^{-1}(k) = \{\mathbf{x} \in \mathcal{T}_{\text{train}} \mid y(\mathbf{x}) = k\}$ denotes the set of all training trajectories belonging to class k. Entry $D_{ij}^k = 1$ indicates that trajectory $\mathbf{x}_i$ from class k exhibits a robustness value for formula φ_j that is distinct from all trajectories in other classes, meaning that φ_j successfully separates $\mathbf{x}_i$ from the rest of the dataset. A formula with high column sum in D^k is thus a strong discriminator for class k.

We seek the smallest set of formulae $\mathcal{F'}_k \subseteq \mathcal{F}_k$ that covers at least a target fraction of class-k trajectories, leading to a minimum-cost set cover problem:

$$\min_{\mathbf{c}} \sum_i c_i \cdot \text{cost}(\varphi_i) \quad \text{s.t.} \quad D^k \mathbf{c} \geq \bar{\gamma}_g |\mathcal{T}_k|, \quad c_i \in \{0, 1\},$$

where $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T}_{\text{train}} \mid y(\mathbf{x}) = k\}$ is the set of class-k training trajectories, $|\mathcal{T}_k|$ is its cardinality, and $\bar{\gamma}_g \in (0, 1]$ specifies the desired coverage fraction (e.g., $\bar{\gamma}_g = 0.9$ requires covering at least 90% of class-k trajectories). Here, $\text{cost}(\varphi_i)$ measures the syntactic complexity of formula φ_i (e.g., node count), and the constraint $D^k \mathbf{c} \geq \bar{\gamma}_g |\mathcal{T}_k|$ ensures that the selected formulae jointly cover at least the target fraction of trajectories, allowing simpler approximate explanations by relaxing the requirement for perfect coverage.

The optimisation is solved via a 0-1 Integer Linear Programme (ILP). If infeasible, a greedy heuristic selects formulae that maximally increase coverage while minimising redundancy.

Global Symbolic Characterisation

The final global explanation for class k is expressed as a disjunction:

$$E_g(k) = \bigvee_{\varphi_i \in \mathcal{F}'_k} \varphi_i,$$

meaning that a trajectory satisfying any of the selected φ_i typically belongs to class k. This provides an interpretable summary of the underlying temporal behaviour of the class.

Crucially, this construction allows the global explanations to be incrementally refined in an online manner: as new trajectories become available, coverage relations can be updated without retraining, ensuring that $E_g(k)$ remains representative of the evolving data distribution.

As in the local case, the final selected formulae undergo logic-aware and data-aware simplification to maximise readability while preserving discriminative and semantic consistency.

Similarly to the local setting, a global importance parameter imp_g is introduced to control the maximum fraction of trajectories that the global explanation is allowed to leave unclassified. By bounding the proportion of points that may be excluded, imp_g explicitly regulates the trade-off between symbolic compactness and global coverage, enabling concise yet representative class-level explanations.

8.5.3 Post-processing and Readability

The STL formulae extracted by the explanation module are semantically faithful but may contain redundancies, overly conservative thresholds, or unnecessary syntactic structure that hinders interpretability. To address this, we apply a comprehensive post-processing pipeline that enhances both readability and discriminative clarity. The procedure operates on three complementary levels: logical simplification, threshold adjustment, and data-aware rewriting. Together, these mechanisms ensure that the final explanations are concise, human-readable, and maintain consistency with the model's learned behaviour. Further details and proofs are provided in Appendix C.

Logical simplification

This component rewrites formulae purely on the basis of their structure, without reference to data. It removes double negations, flattens nested Boolean operators, merges adjacent temporal intervals, and collapses redundant constructs such as

$$\varphi \wedge \varphi \to \varphi, \qquad \mathbf{G}_I(\mathbf{G}_J(\varphi)) \to \mathbf{G}_{I+J}(\varphi).$$

This step systematically reduces structural complexity while maintaining exact semantic equivalence.

Example. The original expression

$$\neg(\mathbf{G}_{[0,20]}(\mathbf{G}_{[5,10]}(x_0 \leq 0.3)))$$

is first simplified structurally to

$$\neg\mathbf{G}_{[5,30]}(x_0 \leq 0.3),$$

and subsequently rewritten, using standard dualities, as

$$\mathbf{F}_{[5,30]}(x_0 > 0.3).$$

This transformation removes nested operators, compresses intervals, and yields a significantly clearer and more intuitive behavioural description.

Threshold adjustment and polarity correction

Extracted formulae may not be perfectly aligned with the robustness landscape observed across classes. To improve discriminative power, we analyse how each formula's robustness varies across the dataset. For a formula φ selected for an input $\mathbf{x}$, let

$r_{\text{target}} = \rho(\varphi, \mathbf{x})$ denote its robustness on the target sample, and let $\mathcal{R}_{\text{opp}}$ be the multiset of robustness values across trajectories belonging to other classes.

If most opposing-class robustness values exceed r_{target}, the polarity of φ is flipped so that the formula becomes positively aligned with the target class. After polarity correction, we adjust the threshold to the midpoint between r_{target} and the closest opposing robustness value below it, thereby enhancing discriminability whilst preserving temporal structure. Thanks to the linearity of robustness with respect to uniform threshold shifts (proved in Appendix C), this procedure is mathematically well-behaved.

Formally, if all predicate thresholds are shifted uniformly by δ, the robustness satisfies $\rho'(\varphi, \tau, t) = \rho(\varphi, \tau, t) - \delta$, for all trajectories τ, times t, and all STL operators. This property ensures that threshold movement alters only the decision boundary, not the temporal semantics of the formula.

Example. Consider a formula $\varphi = \mathbf{G}_{[0,50]}(x_0 \geq 1.2)$ extracted for a Class 0 trajectory with robustness $r_{\text{target}} = 0.35$. Suppose the robustness values on Class 1 trajectories are $\mathcal{R}_{\text{opp}} = \{0.42, 0.51, 0.38, 0.47, \ldots\}$. Since the majority of opposing values exceed r_{target}, the formula incorrectly separates the classes: it is more strongly satisfied by Class 1 than by Class 0. We therefore flip its polarity to obtain

$$\varphi' = \neg\mathbf{G}_{[0,50]}(x_0 \geq 1.2) \equiv \mathbf{F}_{[0,50]}(x_0 < 1.2),$$

which now correctly characterises Class 0 through the eventual violation of the threshold constraint. Next, we identify the closest opposing robustness value below r_{target}, say 0.28, and shift the threshold to their midpoint: $\delta = (0.35 + 0.28)/2 - 0.35 = -0.035$. Applying the uniform shift yields

$$\varphi'' = \mathbf{F}_{[0,50]}(x_0 < 1.165),$$

with adjusted robustness $\rho'(\varphi'', \mathbf{x}) = 0.35 - (-0.035) = 0.385$. This threshold adjustment maximises the margin between target and opposing robustness values, improving the formula's discriminative power whilst preserving its temporal structure.

Data-aware simplification

Beyond structural rewriting, we incorporate empirical information from the dataset. Crucially, this step is performed strictly at the end of the post-processing pipeline, after all logical simplifications and threshold adjustments have been applied. Each atomic predicate is evaluated across all relevant trajectories. Predicates that are always true (resp. always false) are replaced by the Boolean constants $\top$ and $\bot$. Subsequent Boolean simplifications then remove redundant constructs such as

$$\top \wedge \varphi \to \varphi, \qquad \bot \vee \varphi \to \varphi.$$

Temporal operators whose subformulae collapse to constant truth values are simplified accordingly, enabling semantic pruning that cannot be captured by syntactic rules alone.

Example. Consider the formula

$$\varphi = \mathbf{F}_{[0,60]}(x_{\text{vel}} > 10) \wedge \mathbf{G}_{[0,100]}(x_{\text{pres}} < 5.0).$$

Evaluation over the dataset reveals that the pressure signal x_{pres} is physically constrained and never exceeds 4.2 across all observed trajectories. Consequently, the atomic predicate $(x_{\text{pres}} < 5.0)$ evaluates to true for all time points and all samples, allowing us to replace it with $\top$. The formula is thus rewritten as

$$\mathbf{F}_{[0,60]}(x_{\text{vel}} > 10) \wedge \mathbf{G}_{[0,100]}(\top).$$

Recognising that $\mathbf{G}(\top) \equiv \top$ and $\psi \wedge \top \equiv \psi$, the expression collapses to

$$\mathbf{F}_{[0,60]}(x_{\text{vel}} > 10).$$

This reduction eliminates empirically vacuous constraints, ensuring the final explanation focuses solely on the discriminative features that actually vary across the data distribution.

8.5.4 Putting it all together

The entire post-processing pipeline, composed by threshold shift, logical simplification, and data-aware rewriting, is summarised in Algorithm 4, and all simplification rules are listed in Table C.1, both found in Appendix. Each method contributes independently to the removal of redundancies and the sharpening of decision boundaries. The resulting formulae preserve semantic faithfulness to the model's internal reasoning while offering concise, human-readable descriptions of temporal behaviour.

Thus, the final explanations remain both *interpretably symbolic* and *consistent with the empirical structure of the data*, enabling end users to trust and validate the behavioural patterns learned by the model.

Chapter 9

Experimental Evaluation

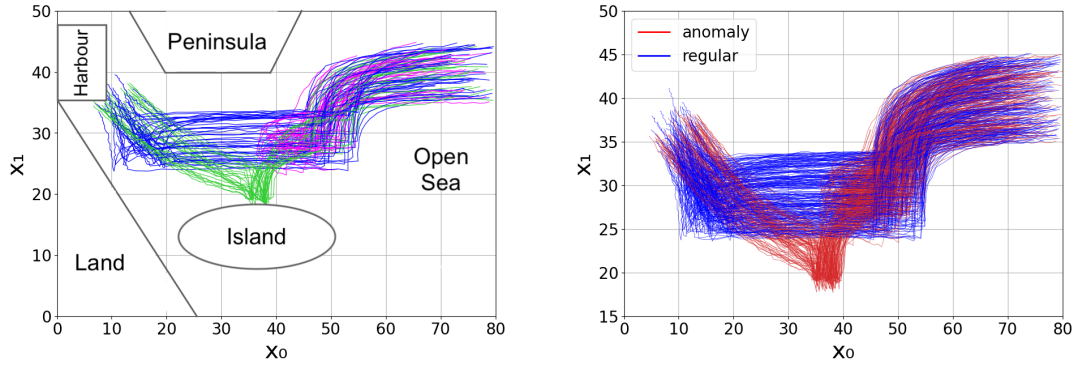
In this chapter, we present an extensive empirical evaluation of STELLE, aimed at assessing both its predictive performance and its interpretability across a diverse range of time series classification tasks. Our experiments are designed to examine three complementary aspects: (i) generalisation across heterogeneous domains, (ii) the capacity to provide semantically meaningful explanations, and (iii) comparative performance with respect to both state-of-the-art data-driven and symbolic baselines.

The selected benchmarks differ substantially in dimensionality, signal length, class cardinality, and temporal structure. They include short univariate physiological traces, high-dimensional multivariate motion recordings, and spatial trajectories arising from a safety-critical maritime scenario. This diversity is intentional: STELLE does not assume stationarity, fixed temporal horizons, homogeneous sampling frequencies, or dataset-specific generative models. Its only structural requirement is that trajectories admit a discrete time index and quantitative STL robustness evaluation. Evaluating the framework across such heterogeneous settings therefore provides a direct empirical test of its claim to structural generality and scale-consistent temporal reasoning.

We begin by introducing the benchmark datasets used in our study, which encompass a variety of domains, signal dimensionalities, and classification complexities. These include large-scale archives such as UEA and UCR, as well as a targeted, real-world maritime surveillance scenario. Following this, we describe the training protocol and experimental setup adopted to ensure reproducibility and fair comparison across methods. Finally, we detail the baseline models employed for quantitative and qualitative comparison, including both modern deep learning methods and interpretable approaches grounded in Signal Temporal Logic (STL).

9.1 Datasets

The selection of datasets aims to capture the diversity of real-world time series scenarios, ranging from benchmark archives widely used in the literature to application-specific domains that highlight STELLE's interpretability. In particular, we include both univariate and multivariate datasets, as well as binary and multiclass problems, to comprehensively assess the model's generality. This combination allows us to evaluate STELLE's capacity to extract temporal logic explanations under varying signal complexity and semantic richness.



(a) (b)

Figure 9.1: (a) Naval surveillance scenario [200], where normal trajectories are shown in blue, and anomalous signals are shown in green and magenta. (b) Examples of trajectories from the naval surveillance case study. The blue and red trajectories belong to normal and anomalous behaviours, respectively.

Maritime Surveillance Dataset. We evaluate STELLE on a binary, multivariate time series classification task derived from a maritime surveillance scenario described by Kong et al. [200] (see Figure 9.1). In this setting, the goal is to detect anomalous vessel behaviours based on their planar trajectories. Normal vessel trajectories proceed directly from open sea towards the harbour, while anomalous trajectories exhibit one of two behaviours: (i) veering towards a nearby island before reaching the harbour, or (ii) approaching other vessels in the channel between the peninsula and the island before returning to the open sea. The dataset contains 2000 trajectories in total, of which 1000 labelled as normal and 1000 as anomalous. Each trajectory is represented as a two-dimensional signal with 61 timepoints, capturing spatial coordinates over time. This dataset provides a concise yet challenging test bed for assessing temporal logic models in the context of safety-critical anomaly detection.

UCR Univariate Archive. To further evaluate STELLE in settings comparable with interpretable symbolic baselines, we also consider a selection of datasets from the UCR Time Series Classification Archive [201]. The UCR archive contains hundreds of univariate, typically multiclass datasets spanning domains such as electrocardiography, image outlines, sensor data, and simulated signals. Compared with the UEA archive, UCR datasets are univariate and generally of lower dimensionality, which allows for a more direct comparison with symbolic and neuro-symbolic methods that learn temporal logic or rule-based models. This benchmark thus highlights STELLE's interpretability and symbolic reasoning abilities in addition to its predictive accuracy.

For this setting, we curated a compact yet diverse subset of datasets from the UCR archive, focusing on problems that are commonly used to evaluate interpretable and symbolic time series models. In particular, the selected datasets correspond to those employed in the STL-based benchmarks considered in this work (see Section 9.3.2). Their inclusion is therefore directly motivated by comparability with existing symbolic and neuro-symbolic approaches. The datasets cover a broad range of temporal characteristics, including short and long sequences, balanced and highly imbalanced

Table 9.1: Summary of the UCR univariate time series datasets used in the experimental evaluation. The dataset type denotes the domain of the recorded signals.

Dataset	TrainSize	TestSize	Length	#Classes	Type
ECG200	100	100	96	2	ECG
ECG5000	500	4500	140	5	ECG
EOGVerticalSignal	362	362	1250	12	EOG
Epilepsy2	80	11420	178	2	EEG
GunPoint	50	150	150	2	HAR
GunPointOldVersusYoung	135	316	150	2	HAR
NerveDamage	163	41	1500	3	EMG
SharePriceIncrease	965	965	60	2	FINANCIAL

train-test splits, and both binary and multiclass classification tasks. Beyond domains already represented in the multivariate benchmark, this selection introduces additional application areas such as image outline classification (IMAGE), electrooculography (EOG), electromyography (EMG), and financial time series analysis (FINANCIAL). This subset enables a focused evaluation of STELLE under conditions where symbolic reasoning and explanation quality are particularly relevant, while keeping the overall experimental workload manageable. Table 9.1 reports the main characteristics of the datasets considered.

UEA Multivariate Archive. We additionally evaluate STELLE on a subset of multivariate time series classification (MTSC) tasks from the University of East Anglia (UEA) Time Series Archive [202]. This benchmark collection includes over thirty datasets covering a wide range of real-world domains such as physiological monitoring, human motion capture, speech and spectrographic data, and environmental sensor readings. Each dataset varies considerably in dimensionality, sequence length, and number of classes, making it a particularly challenging benchmark for assessing the generality and robustness of time series classifiers. The UEA archive serves as the de facto standard for evaluating MTSC algorithms, given its diversity and the established reporting of standardised accuracy metrics. In this work, it is used to demonstrate STELLE's ability to handle complex, high-dimensional temporal dependencies while maintaining interpretability.

To construct a representative yet computationally tractable benchmark, we selected the first ten datasets in alphabetical order from the archive, excluding *DuckDuckGeese*, which was omitted due to its extremely high dimensionality (over 1000 variables) exceeding the available computational resources.
Table 9.2 summarises the key characteristics of the selected datasets, including their short identifiers, size, sequence length, number of classes, and signal dimensionality. The *Type* column indicates the application domain of each dataset: MOTION (motion capture), ECG (electrocardiography), HAR (Human Activity Recognition), SPECTRO (spectrographic signals) and EEG (electroencephalography).

Table 9.2: Summary of the UEA multivariate time series datasets used in the experimental evaluation. Each dataset is identified by a short two-letter code (ID). The dataset type denotes the domain of the recorded signals.

Dataset	ID	TrainSize	TestSize	Length	#Classes	Type	Channels
ArticularyWordRecognition	AW	275	300	144	25	MOTION	9
AtrialFibrillation	AF	15	15	640	3	ECG	2
BasicMotions	BM	40	40	100	4	HAR	6
Cricket	CR	108	72	1197	12	HAR	6
ERing	ER	30	270	65	6	HAR	4
Epilepsy	EP	137	138	207	4	HAR	3
EthanolConcentration	EC	261	263	1751	4	SPECTRO	3
HandMovementDirection	HD	160	74	400	4	EEG	10
Handwriting	HW	150	850	152	26	HAR	3
Libras	LI	180	180	45	15	HAR	2

9.2 Training Protocol

Following an initial round of hyperparameter tuning performed on a representative subset of datasets, several hyperparameters were fixed globally across all experiments. These include the batch size and the learning rates used for both ε_G and the temperature parameter T, from Equation 8.5 and 8.6, respectively. We also fixed $\bar{\gamma}_t$ for explanation extraction and all parameters involved in concept creation.

For the remaining hyperparameters, due to the highly diverse nature of the datasets, we fine-tuned them individually for each dataset to ensure fairness and optimal performance. All experiments were run on computational clusters with GPU allocations of up to 20 GB and CPU memory allocations ranging from 20 GB to 50 GB, depending on dataset size. All reported performance metrics are averaged over 5 cross-validation folds, with 3 independent random seeds per fold. This protocol is adopted to mitigate the effects of data partitioning and stochastic optimisation, and to provide stable and reproducible performance estimates. Moreover, it matches the evaluation protocol used for the results reported in [158], ensuring fair and directly comparable performance reporting.

All source code and scripts for reproducing our experiments are publicly available at https://github.com/ireneferfo/STELLE.

9.3 Baselines

We evaluate STELLE against two complementary groups of baseline methods. The first group consists of established time series classification approaches that prioritise predictive performance and represent the current state of the art across a range of modelling paradigms. The second group comprises interpretable methods based on Signal Temporal Logic or similar temporal languages, which explicitly aim to learn symbolic temporal descriptions from data. This separation allows us to assess STELLE both in terms of classification accuracy relative to strong non-symbolic competitors and in terms of interpretability relative to logic-based and neuro-symbolic approaches.

9.3.1 Time Series Classification Baselines

To contextualise the performance of STELLE, we compare it with a diverse set of state-of-the-art time series classification (TSC) methods. These cover deep learning, transform-based, distance-based, and symbolic or feature-based approaches, as summarised below.

Deep learning methods. We consider representative neural architectures including InceptionTime [77], ResNet [159], and TapNet [81]. In addition, we include the ensemble-based HIVE-COTE 2.0 components, namely HC1 [203] and HC2 [204], which currently define the state of the art in large-scale TSC. While such ensembles achieve exceptional accuracy, they typically rely on hundreds of heterogeneous classifiers whose aggregated decisions reduce transparency, and are therefore not considered intrinsically interpretable or rule-based.

Transform-based methods. These include ROCKET [205], Arsenal, and DrCIF [204], which employ random convolutional kernels and interval-based feature transformations to achieve both scalability and accuracy. ROCKET extracts thousands of features from randomly generated convolutional kernels, while DrCIF extends interval-based features; although both improve expressiveness, the stochastic generation of features limits interpretability. Such methods are well-suited for large archives like UEA due to their efficiency and robustness.

Distance-based methods. We include the canonical nearest-neighbour classifiers using dynamic time warping 1NN-DTW-D, 1NN-DTW-I, and 1NN-DTW-A, which account for dependent, independent, and affine-invariant alignment variants, respectively [189]. These serve as traditional baselines representing interpretable yet non-parametric approaches.

Symbolic and feature-based methods. Finally, we compare against symbolic and interval-feature-based classifiers such as TSF [206], CIF [207], TDE [208], STC [197], MrSEQL [209], MUSE [210], cBOSS [211], and RSF [212]. STC applies Rotation Forest to shapelet features, improving accuracy at the cost of interpretability; RSF and related methods extend feature sets to capture more complex temporal dependencies (e.g., through randomised dilations in RDST), but such randomisation reduces semantic clarity. These approaches provide a degree of interpretability through symbolic representations or engineered interval features, though they lack explicit temporal-logical semantics: their representations do not directly express temporal properties such as ordering, duration, or persistence in a formally grounded language.

For all UEA datasets, we use the accuracy and performance metrics reported in Ruiz et al. [158], which provides the most extensive benchmark currently available for the multivariate archive.

9.3.2 STL-Based Baselines

In addition to standard TSC methods, we compare STELLE against several interpretable approaches based on *Signal Temporal Logic* (STL). These methods aim to derive symbolic,

human-readable formulae that capture temporal patterns directly from data.

Early contributions such as LoTuS by Bombara and Belta [213] introduced a decision-tree-based algorithm for learning STL formulae both offline and online. LoTuS iteratively partitions the input space using STL predicates, producing a tree whose leaves correspond to interpretable temporal logic rules. Although primarily demonstrated on maritime trajectory monitoring tasks, its general framework set the foundation for subsequent STL learning systems.

Building on this foundation, Aasi et al. [214] proposed BCDT, a boosted ensemble of STL decision trees that improves predictive performance through weighted voting while retaining logical interpretability. BCDT enhances robustness and expressivity compared to single-tree models, though at the cost of reduced transparency due to ensemble complexity. Later, Aasi et al. [215] extended this paradigm with an incremental learning strategy, enabling STL classifiers to update their formulae as new temporal segments arrive—an important step toward continual and streaming TSC scenarios.

Evolutionary approaches have also emerged, most notably BUSTLE [216], which formulates STL synthesis as an optimisation problem solved via evolutionary search. BUSTLE discovers temporal specifications that maximise classification accuracy or robustness on labelled trajectories, offering greater flexibility than tree-based models and supporting more complex STL operators. However, evolutionary methods are typically computationally intensive and may require domain-specific fitness functions.

In parallel, Li et al. [114] proposed NN-TLI, a hybrid neuro-symbolic framework that integrates neural networks with STL learning. NN-TLI uses deep encoders to extract features from time series, which are then mapped to STL predicates through differentiable constraints, bridging the gap between data-driven representation learning and logic-based interpretability.

Recent research has further generalised STL learning to multiclass settings. Aguilar et al. [217] presented a specification-mining approach that estimates STL parameters for multiclass classification tasks, effectively extending temporal reasoning beyond binary discrimination. Similarly, Yan et al. [218] introduced NSTSC, a neuro-symbolic model combining convolutional encoders with STL-based reasoning layers to produce global, human-readable formulae on the UCR dataset. Most recently, Wang et al. [219] proposed RRL, which incorporates weighted STL (wSTL) templates into a reinforcement learning setup, enabling rule discovery that balances interpretability and accuracy. Finally, Wang et al. [220] introduced TemporalRule, a modern STL-based classifier that learns reliable and intuitive temporal logic rules using efficient gradient-guided search, achieving state-of-the-art performance on several UCR benchmarks. Most existing models, however, rely predominantly on the *Always* (**G**) and *Eventually* (**F**) operators, neglecting the use of *Until* (**U**) to capture richer temporal dependencies between predicates. This limitation restricts expressiveness and hampers their ability to represent sequential dependencies central to real-world temporal reasoning.

Together, these STL-based baselines illustrate the progression of interpretable time series learning from symbolic rule mining and decision trees to hybrid neuro-symbolic systems. Importantly, all of these methods provide only global, class-level explanations in the form of STL formulae, and do not generate instance-specific explanations for individual trajectories. They therefore provide a meaningful context for assessing STELLE's unique combination of formal semantics, scalability, and explanatory power,

whilst also enabling direct comparison of global explanatory quality.

9.4 Evaluation Metrics

To comprehensively evaluate both predictive performance and interpretability, we employ a set of quantitative metrics spanning classification, explanation efficiency and readability, and computational efficiency.

For predictive performance, we report *accuracy* to characterise the model's behaviour across different types of prediction errors.

For explanation evaluation, we analyse both *local* and *global separability*. For each local explanation $E_\ell(\mathbf{x}, k)$, separability quantifies how well the explanation distinguishes between the target trajectory's class and all other classes. Specifically, for a target trajectory $\mathbf{x}$ we compute:

$$\mathrm{Sep}(E_\ell(\mathbf{x}, k)) = \frac{\#\{y(\tau) \notin k : \mathrm{sign}(\rho(E_\ell(\mathbf{x}, k), \tau)) \neq \mathrm{sign}(\rho(E_\ell(\mathbf{x}, k), \mathbf{x}))\}}{\#\{y(\tau) \notin k\}} \times 100,$$

where $E_\ell(\mathbf{x}, k)$ denotes the local explanation extracted for class k, τ represents trajectories from the training dataset $\mathcal{T}_{train}$, and $y : \mathcal{T} \to \mathbb{N}$ maps trajectories to the target label. This represents the percentage of trajectories from the opposite class whose robustness sign differs from that of the target trajectory, where τ denotes a trajectory from the set of all trajectories not belonging to class k. A high separability value indicates that the explanation generalises well to unseen samples of the same class and excludes those from other classes.

We report the mean standard deviation of local separability under different filtering conditions: (i) only for correctly classified samples (reflecting explanation reliability), (ii) only for misclassified samples, analysed from two distinct perspectives: explanations generated with respect to the true class and with respect to the predicted class. Explanations generated with respect to the true class reveal whether the model's reasoning aligns with actual class characteristics, measuring how well the explanation captures the true class's distinguishing features. Low separability here indicates a misalignment between the model's reasoning and ground truth. On the other hand, explanations generated with respect to the predicted class expose the flawed logic used to justify incorrect decisions, revealing what misleading patterns or correlations the model relies on when making errors. Analysing both perspectives provides comprehensive insight into explanation failure modes and model robustness.

For global explanations, we compute class separability both per-class and overall. The *per-class separability* for class k is given by:

$$\mathrm{Sep}_k(E_g(k)) = \frac{TP_k + TN_k}{\mathrm{total}_k},$$

where TP_k and TN_k respectively denote trajectories of class k that satisfy the class explanation $E_g(k)$ and trajectories of other classes that do not satisfy it. This separability score is closely related to metrics used in prior STL-based classification work: in the binary case, it corresponds exactly to the complement of the misclassification rate

(MCR), that is, $\mathrm{Sep}_k = 1 - \mathrm{MCR}_k$. We therefore interpret separability as the proportion of correctly handled trajectories under the class-level explanation.

The *overall separability* is obtained by micro-averaging across all classes:

$$\mathrm{Sep} = \frac{\sum_k (TP_k + TN_k)}{\sum_k \mathrm{total}_k},$$

representing the total proportion of trajectories correctly classified by their respective class explanations.

We evaluate both measures under three conditions: the entire test set, only correctly classified samples, and only misclassified samples. This allows us to analyse how consistently the explanations reflect the model's decisions.

Finally, to assess interpretability and efficiency, we measure *readability* in terms of the number of syntactic nodes per formula. Together, these metrics provide a balanced assessment of both model performance and the interpretability-efficiency trade-off.

9.5 Analysis on Maritime Surveillance Dataset

We evaluate the classification performance of STELLE across all datasets considered in this study, including the maritime monitoring dataset, univariate benchmarks from the UCR archive, and multivariate datasets from the UEA archive. The objective of this analysis is to assess whether the proposed architecture maintains robust predictive accuracy across a wide range of temporal settings, spanning low-dimensional univariate signals and high-dimensional multivariate trajectories characterised by complex temporal dependencies. Performance is reported using standard accuracy metrics and is compared against relevant baselines to contextualise STELLE's predictive behaviour. All results reported in this section correspond to mean performance over 5 folds and 3 random seeds.

Beyond predictive accuracy, we assess the explanatory behaviour of STELLE in scenarios where the extracted explanations can be meaningfully interpreted and compared. This evaluation focuses on the maritime multivariate dataset and on the univariate datasets from the UCR archive, where domain knowledge and signal characteristics allow for qualitative and quantitative analysis of symbolic explanations. We compare global explanations with those produced by STL-based baselines, as these methods provide only class-level characterisations and do not support instance-specific explanations. Additionally, we report results for the UEA multivariate dataset to demonstrate scalability, though no direct interpretable comparison is provided as the selected STL baselines do not report results for these benchmarks.

For the maritime surveillance task, STELLE achieves near-perfect classification performance, with an average accuracy of 99.7% across five folds and three random seeds. For clarity, we recall that classification accuracy and misclassification rate (MCR) are complementary metrics, related by Accuracy $= 1 -$ MCR; consequently, a zero MCR corresponds to 100% accuracy, while non-zero MCR values indicate imperfect separation. Throughout this section, accuracy refers to the predictive performance of a trained classifier, whereas MCR denotes the error rate induced by directly applying a symbolic specification to partition the dataset.

Table 9.3: Reported results of STL-based baselines and STELLE on the maritime surveillance dataset. Values are quoted directly from the original publications. Accuracy values correspond to predictive models (where applicable), while MCR reflects rule-induced misclassification. Runtimes are approximate averages as reported in the cited works.

Method	Accuracy (%)	MCR (%)	Runtime (min)
STELLE	99.7 ± 0.2	0.125	35 ± 2.3
BUSTLE [216]	–	0.0	6.2
BCDT [214]	–	0.01	33
NN-TLI [114]	–	0.0	0.2
Aguilar et al. [217]	–	0.008/0.002/0.006*	172 ± 0.05

* MCR values reported per class, respectively Class 0, 1 and 2, where Classes 1 and 2 refer to the two types of anomalies present in the dataset.

Table 9.3 summarises the results of representative STL-based baselines as reported in their original publications. Because these approaches were evaluated on distinct datasets and experimental protocols, the listed values should be interpreted qualitatively, illustrating the typical performance and computational characteristics of symbolic temporal-logic learners rather than serving as directly comparable metrics.

Aasi et al. [214] reported that BCDT achieved training and test misclassification rates below 0.01% across several time series tasks, while retaining interpretability through boosted STL ensembles. BUSTLE [216] achieved perfect or near-perfect class separation (MCR $= 0\%$) in benchmark datasets, with moderate runtime on the order of a few minutes. The neural hybrid NN-TLI [114] obtained comparable separation with negligible MCR and minimal runtime (≈ 0.2 min per model). The multiclass framework of Aguilar et al. [217] similarly reported per-class misclassification rates below 0.1%, albeit with substantially higher computational cost (≈ 172 min), due to exhaustive parameter-space exploration.

Overall, these published results confirm that existing STL-based classifiers achieve near-perfect symbolic separation on this maritime benchmark. However, as these models perform rule-based rather than learned classification, their MCR values are not directly comparable to the predictive accuracy achieved by STELLE. We therefore regard them mainly as qualitative baselines to compare STELLE's interpretable rule synthesis.

9.5.1 Explanatory Performance

Figure 9.2 visualises the maritime surveillance dataset by plotting the two recorded variables as functions of time across all trajectories. Presenting the data in the temporal domain highlights characteristic evolution patterns and class-dependent behaviours that are less evident in the standard x_0–x_1 projection typically used to introduce the dataset, depicted in Figure9.1. This representation provides a more suitable context for interpreting temporal logic explanations, as it allows the extracted STL formulae to be directly inspected against the underlying time-resolved signal dynamics.

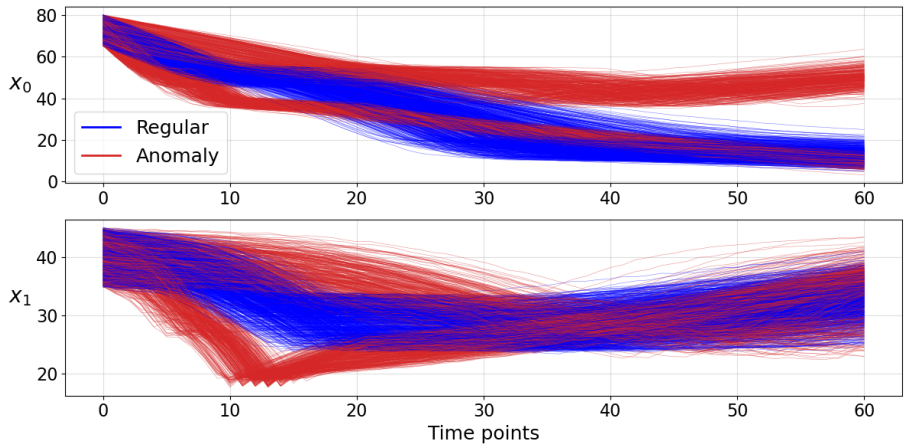


Figure 9.2: Maritime surveillance dataset visualised over time. Each trajectory is shown as a function of time for both recorded variables, illustrating the temporal evolution of normal and anomalous behaviours.

Local explanations

We now analyse instance-level explanations produced by STELLE on the maritime surveillance dataset. These explanations associate individual predictions with compact STL formulae that characterise how a specific trajectory differs from those of the opposite class.

Figure 9.3 reports a representative local explanation for a trajectory predicted as normal, extracted using an importance threshold $\text{imp}_\ell = 0.1$. An identical explanation is obtained when no relevance threshold is applied ($\text{imp}_\ell = 0.0$), we therefore report only one instance, as thresholding does not affect the selected formula nor its discriminative behaviour in this case. The target trajectory is shown in blue, overlaid on the trajectories from the other class in the dataset, and is explained by the STL formula

$$\mathbf{G}\big((x_0 \geq 59.47) \ \vee \ \mathbf{F}_{[16,38]}(x_0 \leq 13.06)\big).$$

This formula captures a global temporal constraint on the primary signal x_0, stating that the trajectory either remains above a high baseline throughout its evolution, or exhibits a pronounced drop within a specific temporal window. Both behaviours are characteristic of regular vessel motion in this dataset and effectively distinguish the target trajectory from anomalous patterns.

For the anomalous class, Figure 9.4 presents two representative local explanations, each corresponding to a distinct anomaly pattern observed in the dataset. These behaviours are captured by the following STL formulae:

$$\neg\big(x_0 \geq 37.92 \ \mathbf{U}_{[13,35]} \ x_0 \leq 60.89\big),$$
$$\neg\big(x_0 \geq 13.42 \ \mathbf{U} \ x_0 \leq 32.75\big).$$

Crucially, both explanations take the form of a *negated* temporal transition. In the maritime context, this signifies a failure to execute a standard navigation manoeuvre. While a regular vessel is expected to traverse specific spatial corridors in sequence (e.g., proceeding from an outer zone to an inner channel), these formulae detect trajectories

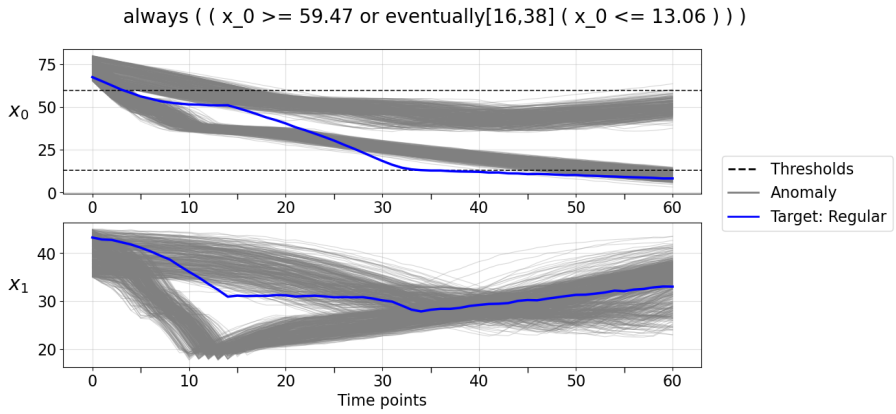


Figure 9.3: Local explanation for a representative normal trajectory in the maritime dataset. The target trajectory (blue) is shown together with the trajectories from the other class (grey), along with the corresponding STL explanation. Dashed lines indicate the thresholds appearing in the formula.

that violate this protocol, either by deviating from the required path before reaching the next checkpoint, or by failing to complete the transit within the expected time window. To understand the discriminative mechanism, consider the positive counterpart of the first formula: $(x_0 \geq 37.92 \ \mathbf{U}_{[13,35]} \ x_0 \leq 60.89)$. This pattern describes normal vessel behaviour: the signal maintains $x_0 \geq 37.92$ from the start, and at some point within the time interval $[13, 35]$, transitions to satisfy $x_0 \leq 60.89$, representing a controlled spatial transition executed within the expected temporal window. For normal trajectories, the precondition $x_0 \geq 37.92$ holds within the considered time interval, after which the vessel enters the target corridor where $x_0 \leq 60.89$. In contrast, anomalous trajectories violate this pattern by transitioning prematurely. The first condition ceases to hold before time step 13, meaning the vessel deviates from the required spatial zone too early, and consequently $x_0 \leq 60.89$ is not satisfied within the prescribed interval. The negation thus captures precisely this temporal violation: anomalies are characterised not by the absence of a transition, but by its occurrence outside the normative time window.
Although structurally similar, the two formulae operate over different spatial ranges and temporal scopes, identifying distinct types of infractions (e.g., a "stalled" approach versus a spatial deviation). This illustrates the ability of STELLE to generate instance-specific explanations that adapt to different failure modes, rather than relying on a single global characterisation of anomalies.
We acknowledge that the specific threshold values (e.g., 37.92, 60.89), jointly with the formulae's structure, could be simplified for improved readability. Making these parameters learnable during training represents a promising direction for future work that could yield more human-friendly formulae whilst preserving discriminative accuracy.

Both explanations identify violations of expected ordered temporal transitions in the signal x_0, corresponding to abnormal changes in vessel behaviour. Although structurally similar, the two formulae operate over different value ranges and temporal scopes, reflecting heterogeneous anomalous dynamics within the same class. This illustrates the ability of STELLE to generate instance-specific explanations that adapt to different failure modes rather than relying on a single global characterisation of anomalies.

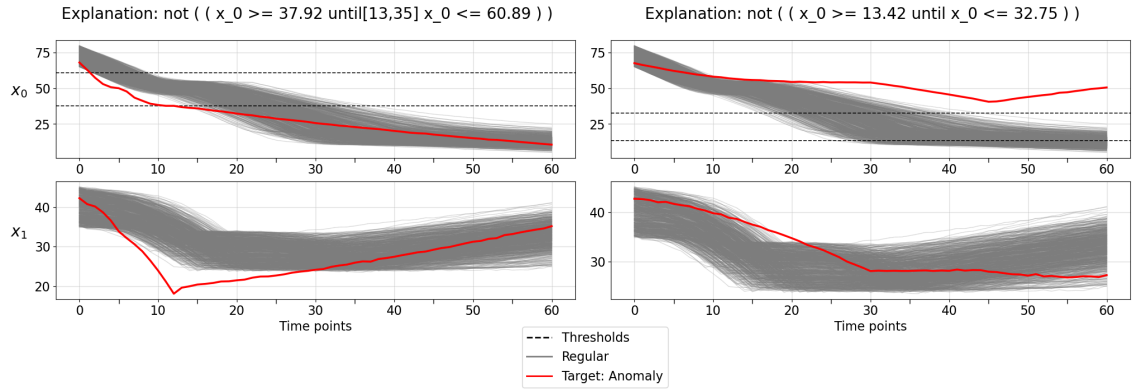


Figure 9.4: Local explanations for two representative anomalous trajectories in the maritime dataset. Each subplot corresponds to a different anomaly pattern and is associated with a distinct STL explanation capturing abnormal temporal behaviour in the primary signal.

Table 9.4: Local explanation quality and complexity metrics. Values show mean $\pm$ std over all test instances.

Class	N	Nodes	Separation (%)	Perfect Sep.	High Quality
0	200	5.84 ± 2.35	99.3 ± 5.0	188 (94.0%)	198 (97.7%)
1	198	4.22 ± 1.14	100.0 ± 0.0	198 (100.0%)	198 (100.0%)

Perfect separation: $= 100\%$; high quality: $\geq 90\%$ separation.

Overall, the extracted formulae highlight interpretable temporal relations between the underlying signals and remain compact enough to be visually inspected alongside the trajectories. These examples demonstrate how STELLE provides faithful, instance-level symbolic explanations that expose the temporal mechanisms driving individual predictions, capabilities unavailable in class-level STL learning approaches. Additional examples of local explanations are reported in Appendix D.

Local explanation analysis. We extracted local explanations for all test instances across both classes, obtaining 200 explanations for Class 0 (Regular) and 198 for Class 1 (Anomaly). The explanations exhibit exceptional discriminative power: 94% of Class 0 explanations and 100% of Class 1 explanations achieve perfect separation ($Sep = 100\%$), indicating that the identified temporal patterns completely distinguish their respective classes from all others. Nearly all explanations (97.7% for Class 0, 100% for Class 1) meet the high-quality threshold of 90% separation (Table 9.4). The 12 Class 0 explanations below this threshold correspond to misclassified instances, where lower separation reflects prediction uncertainty.

Structural diversity analysis reveals that the two classes are distinguished by fundamentally different temporal behaviours, as reported in Table 9.5. Class 0 explanations exhibit moderate structural diversity (16.0% unique structures), with the most common patterns being: (1) persistent bounded oscillations $\mathbf{G}_{[t,\infty]}(x \geq$

Table 9.5: Most common temporal patterns in local explanations. Each pattern represents a distinct class-characteristic behaviour.

Class	Pattern Structure	Count	%
0	Persistent bounded oscillation: $\mathbf{G}_{[t,\infty]}\big(x \geq a \ \wedge \ \mathbf{G}_{[s,u]}(x \leq b)\big)$	37	18.5
	Eventual crossing with constraint: $\mathbf{F}_{[t,\infty]}\big(x \leq a \ \mathbf{U}_{[s,u]} \ x \geq b\big)$	36	18.0
	Global disjunctive constraint: $\mathbf{G}\big((x \geq a) \ \vee \ \mathbf{F}_{[s,u]}(x \leq b)\big)$	33	16.5
	Bounded range constraint: $\mathbf{G}(x \leq a \ \wedge x \geq b)$	32	16.0
1	Forbidden unbounded ordering: $\neg \big(x \geq a \ \mathbf{U} \ x \leq b\big)$	107	54.0
	Forbidden bounded ordering: $\neg \big(x \geq a \ \mathbf{U}_{[s,u]} \ x \leq b\big)$	82	41.4

$a \ \wedge \ \mathbf{G}_{[s,u]}(x \leq b)\big)$, accounting for 18.5% of explanations; (2) eventual threshold crossings with temporal constraints $\mathbf{F}_{[t,\infty]}\big(x \leq a \ \mathbf{U}_{[s,u]} \ x \geq b\big)$, 18.0%; and (3) global disjunctive constraints $\mathbf{G}\big((x \geq a) \ \vee \ \mathbf{F}_{[s,u]}(x \leq b)\big)$, 16.5%. In contrast, Class 1 explanations show remarkable structural concentration (3.5% unique structures), with 95.4% conforming to just two negated *Until* patterns that forbid specific temporal orderings: $\neg \big(x \geq a \ \mathbf{U} \ x \leq b\big)$ (54.0%) and its bounded variant $\neg \big(x \geq a \ \mathbf{U}_{[s,u]} \ x \leq b\big)$ (41.4%). This asymmetry reveals a fundamental difference in how the two classes are distinguished: Class 0 exhibits diverse positive temporal patterns involving persistence, transitions, and disjunctive constraints, whilst Class 1 is predominantly characterised through the negation of specific temporal orderings (*Until* patterns), suggesting that Class 1 trajectories deviate from sequential behaviours expected in standard navigation protocols.

Explanation complexity remains low across both classes, with mean node counts of 5.84 ± 2.35 for Class 0 and 4.22 ± 1.14 for Class 1. This difference is statistically significant (Welch's t-test, $t = 8.72, p < 0.001$), reflecting the greater structural diversity of Class 0 patterns. Post-processing did not reduce complexity in this case (0% reduction), indicating that the generated explanations were already in simplified form. The compactness of these formulae ensures immediate interpretability by domain experts whilst maintaining perfect discriminative power. Although separation percentages differ slightly between classes (99.3% vs 100%), this difference is not statistically significant (Welch's t-test, $p = 0.067$), confirming that both classes achieve equivalently strong discrimination.

Global explanations

We next consider class-level explanations obtained by aggregating concept relevance across trajectories. Table 9.6 reports global explanations for each class under two relevance thresholding strategies: (i) $\text{imp}_g = \text{imp}_\ell = 0$, and (ii) $\text{imp}_g = \text{imp}_\ell = 0.1$. When $\text{imp} = 0$, no relevance-based filtering is applied and all concepts with non-zero relevance are retained, yielding more complex STL specifications that combine a larger number of temporal constraints. In contrast, enforcing a positive threshold retains only

Table 9.6: Global STL explanations for the maritime dataset under different relevance thresholds. Separation indicates the percentage of trajectories correctly separated by robustness sign.

Class	imp*	Sep (%)	Nodes	STL formula
0	0.0	98.5	13	$(x_0 \geq 41.58\ \mathbf{U}\ \mathbf{G}_{[8,25]}(x_0 \leq 33.7)) \vee \big((x_0 \geq 40.13\ \mathbf{U}_{[13,35]}\ x_0 \leq 56.68) \wedge \mathbf{F}(x_0 \leq 29.67) \wedge x_0 \geq 64.33\big)$
0	0.1	99.75	8	$\big((x_0 \geq 40.13\ \mathbf{U}_{[13,35]}\ x_0 \leq 58.68) \wedge \mathbf{F}(x_0 \leq 29.67)\big) \wedge x_0 \geq 64.33$
1	0.0	100.0	12	$\mathbf{F}(\mathbf{G}(x_0 \geq 31.2)) \vee \big(\neg(x_0 \geq 47.01\ \mathbf{U}_{[13,35]}\ x_0 \leq 51.8) \wedge (x_1 \geq 21.67\ \mathbf{U}_{[9,14]}\ x_1 \leq 26.55)\big)$
1	0.1	100.0	12	$\mathbf{F}(\mathbf{G}(x_0 \geq 31.2)) \vee \big(\neg(x_0 \geq 47.01\ \mathbf{U}_{[13,35]}\ x_0 \leq 51.8) \wedge (x_1 \geq 21.67\ \mathbf{U}_{[9,14]}\ x_1 \leq 26.55)\big)$

* $\text{imp} = \text{imp}_\ell = \text{imp}_g$.

the most influential concepts, resulting in more compact explanations.

Quantitatively, both settings achieve very strong class separation in terms of robustness values. With $\text{imp} = 0$, global explanations consist of larger formulae (up to 13 nodes) and achieve separation percentages close to perfect (e.g., 98.5% for class 0 and 100% for class 1). When a positive threshold $\text{imp} = 0.1$ is applied, the resulting explanations are substantially simpler, reducing formula size from 13 to 8 nodes for class 0, while maintaining, and in some cases improving, separation (up to 99.75% and 100%, respectively).

The fact that thresholded explanations can yield equal or slightly better separation than unfiltered ones is not contradictory. When all relevant concepts are retained, the resulting conjunctions and disjunctions may include weakly informative or redundant temporal patterns whose robustness contributions partially cancel out across trajectories. Removing low-relevance concepts acts as a form of structural regularisation, sharpening the dominant temporal relations that consistently discriminate between classes. As a result, relevance thresholding improves readability and robustness without substantially compromising, and occasionally even enhancing, discriminative power.

Overall, these results highlight a clear trade-off between explanatory completeness and interpretability. While $\text{imp} = 0$ preserves the full set of learned temporal constraints, enforcing a modest relevance threshold yields explanations that are markedly easier to inspect and reason about, while remaining strongly aligned with the underlying class separation. Crucially, the fact that these global formulae achieve high separation rates is in line with the model's predictive accuracy (99.7%) confirms that STELLE's high performance is not a "black box" artifact, but is directly grounded in the verifiably distinct temporal patterns captured by the explanations.

Table 9.7: Global explanations extracted by STL-based baselines for the maritime dataset. Some methods provide a single formula that is satisfied by the normal trajectories and violated by the others, whilst others, including STELLE, produce separate per-class characterisations. Note that Aguilar et al. [217] treat the dataset as a three-class problem (distinguishing between two anomaly types); the explanations for both anomaly classes are here combined via disjunction ($\vee$) to enable direct comparison with the binary formulation used by other methods.

Model	Rule
BCDT [214]	$\mathbf{F}_{[28,53]}(x_0 \leq 30.85) \wedge \mathbf{G}_{[2,26]}\big((x_1 > 21.31) \wedge (x_0 > 11.10)\big)$
BUSTLE [216]	$\mathbf{G}_{[11,60]}\big((22.04 < x_1 < 40.74) \wedge (4.01 < x_0 < 56.26)\big)$
NN-TLI [114]	$\mathbf{G}_{[9,14]}(x_1 > 23.37) \wedge \mathbf{F}_{[60,60]}(x_0 < 27.96)$
Aguilar et al. [217]	regular: $(x_1 > 22.44)\ \mathbf{U}_{[40,245]}\ (x_0 \leq 27.37)$ anomaly: $\big((x_1 > 20.63)\ \mathbf{U}_{[40,75]}\ (x_0 \leq 42)\big) \vee$ $\big((x_1 > 29.69)\ \mathbf{U}_{[110,130]}\ (x_0 \leq 61.5)\big)$

Comparison with STL-based baselines

Table 9.7 reports the class-level STL explanations learned by existing STL-based methods, including BCDT, BUSTLE, NN-TLI, and the approach of Aguilar et al. [217], together with the global explanations produced by STELLE. All baseline methods report misclassification rates close to zero on this dataset, indicating strong predictive performance at the symbolic level.

However, a direct semantic comparison between these specifications and those learned by STELLE requires careful consideration of how temporal parameters are treated. In particular, the STL formulae reported by Aguilar et al. [217] include temporal intervals (e.g., $[40, 245]$) that exceed the effective signal horizon of the dataset (61 samples). Such intervals may arise from unconstrained temporal parameter optimisation or implicit time-scale normalisation, and, while not detrimental to classification accuracy, they make the resulting temporal semantics difficult to interpret with respect to the observed trajectories.

More broadly, baseline methods tend to optimise a single compact specification directly for classification performance. By contrast, STELLE constructs global explanations by aggregating relevance-weighted concepts learned during training. This design choice can yield explanations that are slightly longer, as they explicitly combine multiple interpretable temporal patterns to account for heterogeneous behaviours within each class. Importantly, this additional structure reflects explanatory completeness rather than unnecessary complexity.

9.6 Analysis on Univariate UCR Datasets

The set of STL-based baselines considered for the UCR univariate benchmarks differs from those used in the maritime case study. This discrepancy is due to the availability of published results rather than a selective evaluation choice. In particular, RRL,

Table 9.8: Classification accuracy comparison between STELLE and interpretable STL-based baselines on a subset of UCR datasets. The column "Mean$_{STL}$" indicates the average performance of all baseline classifiers for each dataset, providing a reference for comparative ranking. "Mean$_{bo}$" reports the average accuracy of common non-interpretable baselines (HC2, InceptionTime, ROCKET, etc.) from the 2024 UCR bakeoff [221]; cells are left empty for datasets not included in that benchmark. "TR" abbreviates TemporalRule. STL baseline results are taken from Wang et al. [220]. For clarity, the best result for each dataset and overall (excluding the mean columns) is shown in bold.

Dataset	RRL	NSTSC	TR	Mean$_{STL}$	Mean$_{bo}$	**STELLE**
ECG200	0.78	0.87	**0.89**	0.85	0.87	0.8
ECG5000	0.92	0.93	**0.94**	0.93	0.92	0.92
EOGVerticalSignal	0.25	0.37	0.46	0.36	0.69	**0.58**
Epilepsy2	0.75	0.93	**0.95**	0.87	-	**0.95**
GunPoint	0.81	0.96	**1.0**	0.92	0.98	0.92
GunPointOldVersusYoung	**1.0**	**1.0**	**1.0**	1.0	0.99	**1.0**
NerveDamage	0.46	**1.0**	**1.0**	0.82	-	**1.0**
SharePriceIncrease	0.59	0.66	**0.69**	0.65	-	0.62
Mean	0.70	0.84	**0.87**	0.80	0.89*	0.85

* The overall mean for Mean$_{bo}$ is computed over the available datasets only.

NSTSC, and TemporalRule report per-dataset classification accuracy on the specific UCR benchmarks considered here, as documented in Wang et al. [220], enabling a direct and fair comparison under matched experimental conditions.
By contrast, other interpretable methods evaluated in the multivariate maritime setting, including BUSTLE, BCDT, NNLI, and the approach of Aguilar et al. [217], do not provide per-dataset results for these UCR benchmarks in their original publications. In the case of Aguilar et al. [217], only cumulative performance metrics over the UCR archive are reported, without dataset-level breakdowns, which prevents meaningful inclusion in Table 9.8.
Re-implementing or re-evaluating these methods outside their originally reported settings would introduce additional sources of variability and potentially compromise comparability. We therefore restrict each experimental comparison to baselines for which authoritative, dataset-specific results are available, following established practice in the literature.

To contextualise these results, we introduce an aggregate reference score that is not intended as a direct baseline comparison. Specifically, an additional column, Mean$_{bo}$, reports the average accuracy of standard non-interpretable baselines (e.g., HC2, InceptionTime, ROCKET) from the 2024 UCR bakeoff [221]. This provides an approximate upper-bound performance reference for each dataset, without implying experimental equivalence.
For three datasets (*Epilepsy2*, *NerveDamage*, and *SharePriceIncrease*), these values are unavailable, as the datasets were excluded from the 2023 bakeoff due to deprecation or quality filtering in later UCR releases. Nevertheless, they are retained here as they are commonly used in prior STL-based studies, including those defining the baselines

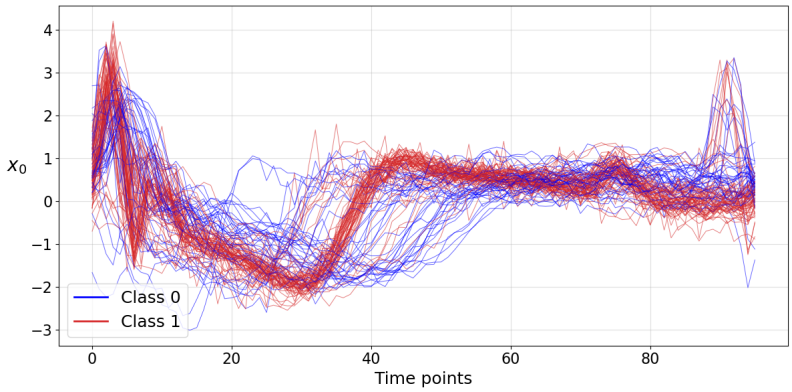


Figure 9.5: Visualisation of the *ECG200* dataset. Each time series traces the electrical activity of a single heartbeat, classified as either Normal or Myocardial Infarction. Unlike standard projections, presenting the data in the temporal domain highlights the evolution patterns and characteristic deviations in wave shape and timing targeted by the STL explanations.

considered, thus preserving comparability with earlier interpretable approaches.

Overall, STELLE achieves competitive performance across all datasets, with a mean accuracy of 0.85, closely aligned with the average performance of the interpretable baselines (0.87) and within a small margin of the non-interpretable bakeoff mean (0.89). On several datasets, STELLE matches the strongest STL-based methods: it achieves perfect accuracy on *GunPointOldVersusYoung* and *NerveDamage*, matching NSTSC and TemporalRule, and reaches 0.95 on *Epilepsy2*, equal to the best reported baseline. On *ECG5000*, STELLE attains 0.92, in line with both the STL baseline mean and the bakeoff average, and only marginally below the best result of 0.94.

On datasets where STELLE does not achieve the highest accuracy, the performance gap remains modest and follows a consistent pattern. Lower accuracy is observed primarily on datasets with higher input dimensionality or weaker temporal regularities, such as *EOGVerticalSignal* and *SharePriceIncrease*. In these settings, discriminative information is distributed across many channels or exhibits substantial stochastic variability, rather than being organised around stable temporal patterns. As a result, it is less amenable to representation through a fixed set of semantically meaningful STL concepts. By contrast, on low-dimensional physiological and motion datasets, where class distinctions are governed by consistent and interpretable temporal behaviours, STELLE matches or exceeds competing interpretable baselines.

These trends are consistent with the design philosophy of STELLE. By constraining concept complexity and favouring shallow, semantically meaningful STL formulae, the framework prioritises interpretability and stability of explanations. In datasets where class separation relies on relatively simple and temporally local patterns, this inductive bias proves effective.

9.6.1 Explanatory Performance

As we did for maritime trajectories, we analyse univariate time series from the UCR archive to demonstrate the versatility of our approach across different domains.

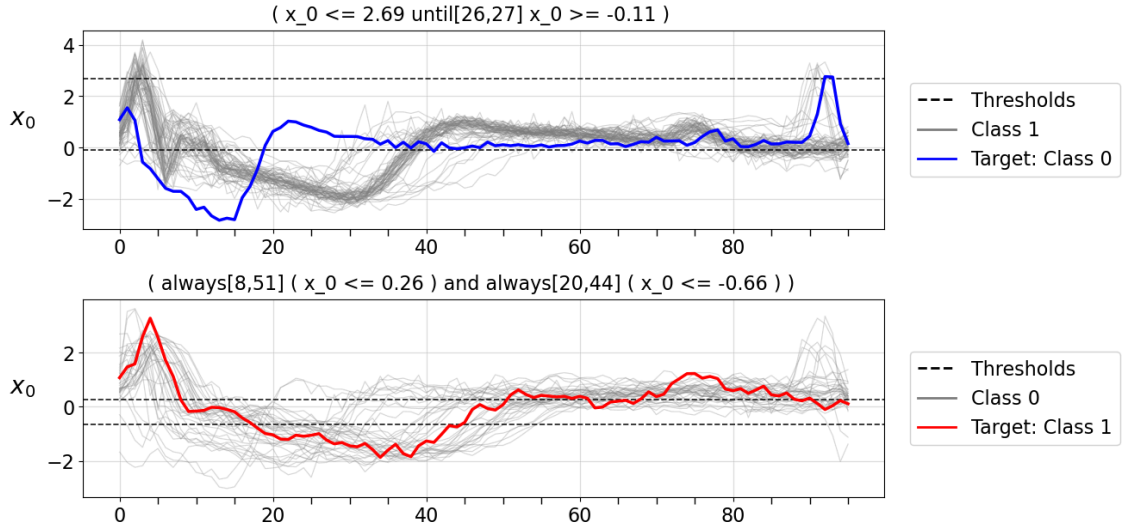


Figure 9.6: Example of local explanations for two correctly classified representative trajectories in the ECG200 dataset. The top panel shows a Normal heartbeat (Class 0, blue) characterised by a specific recovery transition. The bottom panel shows a Myocardial Infarction heartbeat (Class 1, red) characterised by suppressed amplitude constraints.

Figure 9.5 visualises the *ECG200* dataset, which traces the electrical activity of the heart during individual beats. The data distinguishes between normal heartbeats (Class 0) and those indicative of Myocardial Infarction (Class 1). Plotting these signals in the temporal domain highlights the subtle morphological differences, such as variations in peak amplitude and wave duration, that define the Myocardial Infarction condition, providing a direct visual basis for the temporal logic constraints identified by our method.

Local Explanations

We now analyse instance-level explanations produced by STELLE on the ECG200 dataset. These explanations associate individual predictions with compact STL formulae that characterise how a specific heartbeat morphology differs from those of the opposite class. The top panel of Figure 9.6 reports a representative local explanation for a heartbeat classified as Normal (Class 0). The target trajectory is shown in blue, overlaid on the trajectories of the Myocardial Infarction class (grey), and is explained by the STL formula

$$(x_0 \leq 2.69)\ \mathbf{U}_{[26,27]}\ (x_0 \geq -0.11).$$

This formula captures a specific temporal transition in the cardiac cycle. It dictates that the electrical signal must remain bounded ($x_0 \leq 2.69$) until it successfully recovers to a baseline level ($x_0 \geq -0.11$) within the strict temporal window $t \in [26, 27]$. This precise timing of the recovery phase effectively distinguishes the normal rhythm from the Myocardial Infarction beats, which often exhibit delayed or distorted repolarisation patterns.

For the Myocardial Infarction class (Class 1), the bottom panel of Figure 9.6 presents a representative local explanation. The anomalous behaviour is captured by the following

Table 9.9: Local explanation quality and complexity metrics for ECG200. Values show mean ± std. Note the significant drop in separation quality for misclassified instances compared to Class 0 ($p < 0.001$).

Group	N	Nodes	Separation (%)	diversity ratio
Class 0 (Correct)	24	4.00 ± 2.08	99.93 ± 0.24	0.33
Class 1 (Correct)	62	6.71 ± 2.66	97.98 ± 4.28	0.65
Misclassified	14	5.21 ± 2.49	93.24 ± 9.62	0.86

STL formula:

$$\mathbf{G}_{[8,51]}(x_0 \leq 0.26) \wedge \mathbf{G}_{[20,44]}(x_0 \leq -0.66).$$

This explanation identifies the pathology through the *absence* of normal peak activity. It imposes two concurrent global constraints: the signal must remain below 0.26 throughout the broad interval $t \in [8, 51]$, and strictly below -0.66 during the core interval $t \in [20, 44]$. By enforcing these low-amplitude thresholds, the formula characterises the "flattened" morphology typical of the infarcted heartbeat, explicitly distinguishing it from the healthy class which consistently violates these bounds with high-amplitude peaks.

Overall, the extracted formulae highlight interpretable temporal relations, specifically the contrast between the timed transitions of healthy beats and the amplitude suppression of Myocardial Infarction ones. These examples demonstrate how STELLE provides faithful, instance-level symbolic explanations that expose the temporal mechanisms driving individual clinical predictions.

Local explanation analysis. We extracted local explanations for the entire test set of the ECG200 dataset, obtaining 24 explanations for correctly classified Class 0 (Normal) instances, 62 for correctly classified Class 1 (Myocardial Infarction) instances, and 14 for misclassified instances.

As reported in Table 9.9, the explanations for correctly classified samples demonstrate high discriminative validity. Class 0 explanations achieve nearly perfect separation ($99.9 \pm 0.2\%$), whilst Class 1 explanations maintain very high performance ($98.0 \pm 4.3\%$). In contrast, explanations for misclassified trajectories, extracted with reference to the true class, exhibit a significant drop in discriminative quality ($93.2 \pm 9.6\%$). The difference in separation quality between correctly classified Class 0 instances and misclassified ones is statistically significant (Mann-Whitney $U = 248.5, p < 0.001$), confirming that the model's explanatory confidence degrades when prediction reliability is compromised.

Structural diversity analysis (Table 9.10) reveals fundamental differences in how the model characterises the two classes. We quantify this using the *diversity ratio*, defined as the proportion of unique logical structures (ignoring specific numerical thresholds) relative to the total number of explanations; a lower ratio indicates high consistency, whilst a value near 1.0 implies that each instance requires a unique ad-hoc formula. The Normal class (Class 0) shows high structural consistency (diversity ratio 0.33), with 62.5% of explanations conforming to just two patterns: a timed transition $\varphi \mathbf{U}_{[t,s]} \psi$ (37.5%) or a disjunctive amplitude constraint $(x \leq a \vee x \geq b)$ (25.0%). This suggests that healthy

Table 9.10: Most common temporal patterns in local explanations for ECG200. Class 0 is defined by consistent timing and amplitude templates, whereas Class 1 and Misclassified instances are characterised by high diversity and complex nested operators.

Group	Pattern Structure	Count	%
Class 0	Timed transition: $(x \leq a \, \mathbf{U}_{[t_1,t_2]} \, x \geq b)$	9	37.5
	Disjunctive amplitude: $(x \leq a \vee x \geq b)$	6	25.0
	Global bounded range: $\mathbf{G}_{[t_1,t_2]}(x \leq a \wedge x \geq b)$	4	16.7
Class 1	Nested eventuality: $(\mathbf{F}\phi \, \mathbf{U} \, \psi) \wedge \mathbf{G}(\mathbf{F}\xi)$	7	11.3
	Global disjunction: $\mathbf{G}_{[t_1,t_2]}(x \geq a \vee x \leq b)$	4	6.5
	Other unique complex structures	51	82.2
Misclass.	Disjunctive amplitude: $(x \leq a \vee x \geq b)$	3	21.4
	Other unique complex structures	11	78.6

heartbeats are recognised through a predictable template of timing and amplitude variance. Conversely, the Myocardial Infarction class (Class 1) exhibits high structural diversity (diversity ratio 0.65), with 40 unique structures across 62 instances. The most common pattern (11.3%) is a complex nesting of eventualities and global constraints, reflecting the heterogeneous nature of myocardial infarction, which manifests through diverse distortions in wave morphology rather than a single unified signature.
The misclassified instances show the highest structural entropy (diversity ratio 0.86), with 12 unique structures found in just 14 samples. This implies that when the model errors, it fails to identify robust features, instead generating ad-hoc, structurally unique formulae to fit the specific trajectory.

Complexity metrics further validate these findings. Myocardial Infarction (Class 1) explanations are significantly more complex (mean nodes 6.71) than Normal (Class 0) ones (4.00), a difference confirmed by Welch's t-test ($t = -4.93$, $p < 0.001$). This reinforces the observation that distinguishing specific infarction patterns requires more elaborate logic than defining the standard healthy rhythm.

Global explanations

We next consider class-level explanations obtained by aggregating concept relevance across the ECG200 dataset. Unlike the Maritime setting, where anomalies shared consistent patterns, our local analysis revealed high structural diversity in the pathological class. Consequently, we focus our global analysis on the thresholded setting ($\text{imp}_g = \text{imp}_\ell = 0.1$). This strategy is chosen to rigorously filter out the heterogeneous, instance-specific patterns found in the local stage, isolating only those dominant temporal constraints that are universally valid across the entire class.

Table 9.11 reports the extracted global formulae. For the Normal class (Class 0), the method identifies a robust signature characterised by bounded amplitude stability. The

Table 9.11: Global STL explanations for the ECG200 dataset with relevance threshold imp $= 0.1$. The low separation for Class 1 reflects the high structural diversity of anomalies, which prevents aggregation into a single coherent formula.

Class	Sep (%)	Nodes	STL formula
0	85.3	7	$\mathbf{G}_{[20,44]}(x_0 \leq 0.28) \vee \mathbf{G}_{[30,41]}(-1.37 \leq x_0 \leq 2.21)$
1	53.1	4	$x_0 \geq -2.57\ \mathbf{U}_{[27,53]}\ \mathbf{F}_{[38,\infty]}(x_0 \leq 4.68)$

extracted formula is a disjunction of two global constraints:

$$\mathbf{G}_{[20,44]}(x_0 \leq 0.28) \vee \mathbf{G}_{[30,41]}(-1.37 \leq x_0 \leq 2.21).$$

This specification achieves a separation of 85.3%, successfully capturing the regularity of a healthy heartbeat, which must remain within specific amplitude corridors during the critical phases of the cardiac cycle.

In contrast, the global explanation for the Myocardial Infarction class (Class 1) yields a separation of only 53.1%, effectively failing to distinguish the class globally. This result, while quantitatively poor, is structurally instructive. As shown in the local analysis (Table 9.10), pathological heartbeats exhibit extreme structural diversity (82.2% of patterns fall into "other unique complex structures"). Because there is no single "anomalous template" shared across all patients, the aggregation of relevance scores dilutes the disparate local signals. The resulting formula, a generic *Until* property, represents the only weak pattern common to the set, confirming that Class 1 is defined by structural variability rather than the presence of a unified global signature.

Crucially, the contrast between the model's high predictive accuracy and the low separability of the global Class 1 formula demonstrates that STELLE is not limited by the expressiveness of a single global rule. Instead, the architecture successfully leverages its kernel-based embedding to resolve the structural heterogeneity of the anomalous class, achieving accurate classification even when no unified symbolic template exists.

Comparison with STL-based baselines

Table 9.12 reports the global explanations generated by state-of-the-art STL-based methods for the ECG200 dataset. A fundamental distinction lies in the explanation structure. Existing methods such as RRL, NSTSC, and TemporalRule typically formulate the problem as binary classification via a single discriminant rule. They learn one formula φ that characterises the positive class, implicitly defining the negative class simply as $\neg\varphi$. This approach assumes a symmetry in discriminative power that often does not hold in practice, particularly for asymmetric domains where anomalies are structurally diverse. By contrast, STELLE generates independent global explanations for each class, enabling it to reveal when a class is coherent (like Normal heartbeats) versus when it is a heterogeneous collection of errors (like Myocardial Infarction).

Regarding interpretability, a critical limitation of baselines like RRL and NSTSC is their treatment of temporal data. As shown in Table 9.12, these methods generate formulae that are merely logical combinations of predicates at individual, often

Table 9.12: Examples of global explanations extracted by STL-based baselines for the ECG200 dataset. In the formulae generated by TemporalRule [187], x^D indicates the derivative, while x^R indicates the raw values.

Model	Rule
RRL [219]	$(x_{29} > -1.835) \wedge (x_{82} > -0.455) \wedge (x_{83} > -0.305) \wedge (x_{87} > -0.485)$ $\wedge (x_{94} > -0.250) \wedge (x_3 < 3.260) \wedge (x_{44} < 0.824) \wedge (x_{69} < 0.662)$ $\wedge (x_{74} < 0.841) \wedge (x_{75} < 0.872) \wedge (x_{76} < 1.205) \wedge (x_{77} < 0.796)$
NSTSC [180]	$(x_{32} \leq -2.220) \wedge (x_{37} \leq -0.839) \wedge (x_{38} \leq -0.406) \wedge (x_{39} \leq -0.144)$
TemporalRule [220]	$(x^D < 0.16)\ \mathbf{U}_{[22,27]}\ (0.41 \leq x^D < 0.91)$

disconnected, time points (e.g., checking x_{82} and x_{94} independently). Consequently, the resulting explanations are lengthy and fragmented, and they lack inherent temporal continuity. To derive a meaningful temporal property, such as a persistent interval or a trend, one would need to post-process these static checks to manually merge adjacent points with high weights.

While TemporalRule [187] improves upon this by employing STL operators, it presents different interpretability challenges. As seen in Table 9.12, TemporalRule relies on the *Until* operator ($\mathbf{U}$) applied to signal derivatives (x^D). While mathematically expressive, the reliance on derivative thresholds (e.g., $x^D < 0.16$) rather than raw amplitude abstracts the explanation away from the visual morphology of the ECG. Clinicians typically diagnose based on visible shapes (e.g., ST-elevation) rather than rates of change, making derivative-based rules less intuitive for domain experts.

STELLE, conversely, inherently learns temporal operators over continuous intervals (e.g., $\mathbf{G}_{[20,44]}$). It directly captures the duration and evolution of the signal without requiring the user to mentally reconstruct the timeline from scattered time points. This yields compact formulae that operate on the raw signal values and offer immediate clinical interpretation, such as identifying explicit amplitude bounds maintained over a specific phase of the cardiac cycle.

9.7 Analysis on Multivariate UEA Datasets

Table 9.13 reports classification accuracy results on the UEA multivariate archive, comparing STELLE with a broad range of state-of-the-art time series classification methods. For readability, methods are split across two tables, while mean accuracy and global rankings are computed jointly across all approaches.

We do not include a direct comparison with STL-based baselines on the UEA multivariate archive. In this case, we report baseline accuracies from the large-scale benchmark of Ruiz et al. [158] and evaluate STELLE under the same protocol, without re-running symbolic methods. Moreover, most existing STL-based classifiers are limited to binary settings. Only a small subset supports multiclass classification, whereas all UEA datasets considered here are multiclass. For this reason, comparisons with STL-based approaches are restricted to datasets where directly comparable results are available.

Table 9.13: Classification accuracy comparison between STELLE and state-of-the-art time series classification methods on the UEA multivariate archive. For readability, methods are split across two tables, while performance statistics are computed jointly across all approaches. Reported values correspond to classification accuracy, with best results highlighted in bold. The column "Mean" indicates the average accuracy across datasets for each method. The row "Rank" reports the ranking of each method based on mean accuracy, computed across all methods appearing in both tables (ties share the same rank). Abbreviations used are: DTW-D (*1NN-DTW-D*), DTW-I (*1NN-DTW-I*), and DTW-A (*1NN-DTW-A*). Performance metrics are sourced from Ruiz et al. [158].

Dataset	DTW-D	ROCKET	STC	TapNet	HC2	DrCIF	DTW-I	Arsenal	RISE	MrSEQL	**STELLE**
AW	0.99	**1.0**	0.98	0.97	**1.0**	0.98	0.94	**1.0**	0.96	0.99	0.95
AF	0.24	0.25	0.32	0.3	0.28	0.23	0.35	0.26	0.24	0.37	0.36
BM	0.95	0.99	0.98	0.99	0.99	**1.0**	0.72	0.99	**1.0**	0.95	0.99
CR	**1.0**	**1.0**	0.99	0.97	**1.0**	0.99	0.96	**1.0**	0.98	0.99	0.96
ER	0.93	0.98	0.84	0.89	**0.99**	0.98	0.91	0.98	0.82	0.93	0.84
EP	0.96	0.99	0.99	0.96	**1.0**	0.99	0.67	0.99	**1.0**	**1.0**	0.94
EC	0.3	0.45	**0.82**	0.29	0.79	0.67	0.31	0.48	0.49	0.6	0.32
HD	0.3	0.45	0.35	0.32	0.4	0.46	0.27	0.45	0.28	0.35	0.47
HW	0.61	0.57	0.29	0.33	0.56	0.34	0.34	0.55	0.18	0.54	0.2
LI	0.88	0.91	0.84	0.84	0.93	0.91	0.79	0.89	0.82	0.87	0.75
Mean	0.72	0.76	0.74	0.69	0.79	0.76	0.63	0.76	0.68	0.76	0.73
Rank	14	5	10	18	2	5	21	5	20	5	13

Dataset	MUSE	InceptionT	HC1	cBOSS	ResNet	TSF	TDE	DTW-A	CIF	RSF	Mean	**STELLE**
AW	0.99	0.99	0.98	0.98	0.98	0.95	0.98	0.99	0.98	0.98	0.98	0.95
AF	**0.74**	0.22	0.29	0.3	0.36	0.3	0.3	0.22	0.25	0.28	0.31	0.36
BM	**1.0**	**1.0**	**1.0**	0.99	**1.0**	**1.0**	0.99	**1.0**	**1.0**	**1.0**	0.98	0.99
CR	**1.0**	0.99	0.99	0.98	0.99	0.93	0.99	**1.0**	0.98	0.97	0.98	0.96
ER	0.97	0.92	0.94	0.84	0.87	0.9	0.94	0.93	0.96	0.92	0.92	0.84
EP	**1.0**	0.99	**1.0**	**1.0**	0.99	0.97	**1.0**	0.97	0.98	0.96	0.97	0.94
EC	0.49	0.28	0.81	0.4	0.29	0.45	0.53	0.3	0.73	0.34	0.49	0.32
HD	0.38	0.42	0.38	0.29	0.35	0.49	0.38	0.31	**0.52**	0.32	0.37	0.47
HW	0.52	**0.66**	0.5	0.49	0.6	0.36	0.56	0.61	0.35	0.37	0.47	0.2
LI	0.9	0.89	0.9	0.85	**0.94**	0.8	0.88	0.88	0.92	0.76	0.87	0.75
Mean	**0.8**	0.74	0.78	0.71	0.74	0.72	0.76	0.72	0.77	0.69	0.73	0.73
Rank	**1**	10	3	17	10	14	5	14	4	18	–	13

Across several datasets, many methods achieve near-perfect accuracy, particularly on problems such as *ArticularyWordRecognition* (AW), *BasicMotions* (BM), *Cricket* (CR), and *Epilepsy* (EP). These results indicate pronounced ceiling effects, under which differences between classifiers are necessarily small and should be interpreted with caution, as marginal accuracy variations may not correspond to meaningful differences in predictive behaviour.

When considering mean accuracy across all datasets, ensemble- and interval-based classifiers occupy the top ranks, reflecting their strong performance on heterogeneous multivariate time series. Within this global comparison of 21 methods, STELLE achieves a mean accuracy of 0.73, corresponding to a global rank of 13. Its performance is comparable to that of several established shapelet-, interval-, and kernel-based approaches, and exceeds that of simpler distance-based baselines such as DTW-I.

A more detailed analysis reveals that STELLE's performance is closely linked to dataset characteristics. In particular, datasets with very limited training data, such as *AtrialFibrillation* (AF) and *ERing* (ER), pose significant challenges, as learning both discriminative decision boundaries and stable concept relevance patterns requires sufficient supervision. Performance also degrades on datasets with a large number of classes when class-specific temporal dynamics are highly heterogeneous, as observed for *Handwriting* (HW) and *Libras* (LI). In contrast, datasets such as *ArticularyWordRecognition* remain comparatively easier due to stronger class separability and more consistent temporal structure.
Sequence length further influences performance. STELLE tends to achieve stronger results on short-to-medium length time series, while very long and noisy signals, such as those in the *EthanolConcentration* (EC) dataset, are more difficult to capture using a fixed set of temporal logic concepts. Additionally, STELLE performs more robustly on structured motion and activity recognition datasets than on physiological signals, including ECG and EEG data, which usually exhibit substantial inter-subject variability and less consistent temporal patterns.

Overall, instances in which STELLE performs significantly worse than the strongest accuracy-focused methods are not arbitrary, but correspond to specific dataset regimes characterised by limited data, high variability, or complex and heterogeneous dynamics. In contrast, when class distinctions are driven by shared and repeatable temporal relations that can be naturally expressed using symbolic temporal concepts, STELLE remains competitive with established state-of-the-art classifiers.

9.7.1 Explanatory Performance

We extend our analysis to three representative multivariate datasets from the UEA archive: *BasicMotions* (BM), *ERing* (ER), and *Epilepsy* (EP). These datasets were selected to evaluate STELLE under conditions prioritising classification accuracy ($\text{imp}_\ell = \text{imp}_g = 0$).

Local explanations

Table 9.14 reports the complexity of the generated local explanations in terms of syntactic node counts. A striking pattern emerges when distinguishing between correctly classified trajectories and misclassified ones. Explanations for correctly

Table 9.14: Readability statistics of local explanations for UEA datasets (Post-processing). "C" denotes correctly classified samples; "Pred IC" and "True IC" refer to misclassified samples explained wrt the predicted and true labels, respectively. Values are Mean ± Std.

	Syntactic Nodes			Distinct Variables		
Dataset	C	Pred IC	True IC	C	Pred IC	True IC
BasicMotions	9.07 ± 11.2	19.0 ± 11.3	14.3 ± 4.7	2.08	4.00	3.67
ERing	3.78 ± 2.16	4.74 ± 2.32	4.69 ± 3.0	1.30	1.59	1.51
Epilepsy	9.20 ± 9.71	9.67 ± 6.18	34.0 ± 31.9	1.58	2.33	2.00

classified instances (C) remain concise and semantically coherent across all datasets (e.g., 3.78 nodes for ERing). In contrast, explanations for misclassified instances, whether extracted with respect to the predicted (Pred IC) or true (True IC) label, are substantially more complex (e.g., rising to 34.0 nodes for Epilepsy True IC). This observation mirrors the findings in the ECG200 analysis: when the model is confident and correct, it relies on simple, robust temporal features. When it errs, or attempts to justify a class not supported by the data, it resorts to elaborate, high-complexity formulae to fit the internal representation. Post-processing proves effective across the board, reducing formula length without altering the underlying feature set, as evidenced by the stable variable counts.

Figure 9.7 illustrates this capability qualitatively using the *ERing* dataset. The figure displays two representative local explanations. The first identifies a suppression pattern in the first channel:

$$\mathbf{G}(x_0 \leq -1.95),$$

while the second captures a distinct activity spike in the second channel during a specific interval:

$$\mathbf{F}_{[30,40]}\ (x_1 \geq 2.16).$$

These formulae are compact, human-readable, and geometrically faithful, with the target trajectories (red) clearly exhibiting the threshold violations described by the logic.

Global explanations

We next consider the class-level explanations obtained by aggregating these local concepts. Table 9.15 summarises the performance of the global classifiers. The separability scores remain high (e.g., 71.8% for Epilepsy), indicating that the aggregated logic successfully captures discriminative patterns. Notably, the faithfulness metrics are strong: Specificity is consistently high ($> 93\%$) and Precision is robust ($> 80\%$), confirming that the global explanations provide a selective and accurate proxy for the black-box model's decision boundaries.

Figure 9.8 reports the global explanation for Class 2 of the *ERing* dataset. Unlike the atomic local rules, the global formula reveals the multi-modal nature of the class through

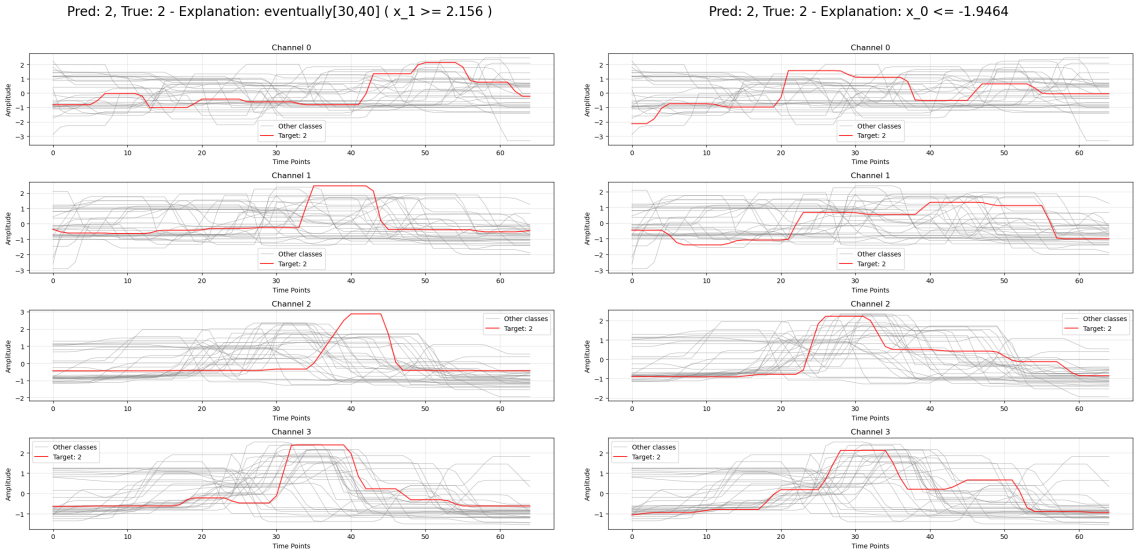


Figure 9.7: Local explanations for two representative trajectories in the *ERing* dataset. The target trajectories (red) are distinguished from the other classes (grey) by clear temporal signatures: a global suppression on channel x_0 (left) and a localised spike on channel x_1 (right).

Table 9.15: Quality of Global Explanations for UEA datasets. "Sep" denotes Separability on all data. Recall, Specificity (Spec), and Precision (Prec) measure faithfulness to the black-box model.

Dataset	Sep (%)	Nodes (Post)	Recall	Spec	Prec
BasicMotions	60.6%	30.2 ± 26.8	50.0%	97.5%	87.0%
ERing	71.4%	8.7 ± 3.2	66.7%	98.7%	98.9%
Epilepsy	71.8%	101.8 ± 52.0	74.6%	93.9%	80.5%

a disjunctive structure:

$$\mathbf{F}_{[12,19]}(x_1 \geq 1.84) \vee$$
$$\big(\mathbf{G}_{[29,38]}(x_2 \leq -0.33) \ \wedge \ \mathbf{G}_{[20,22]}(\mathbf{F}_{[3,33]}(x_3 \leq -0.05) \wedge x_3 \geq 1.56)\big).$$

This formula indicates that there are distinct temporal "pathways" to Class 2: a trajectory may qualify via an early spike in x_1, or through a complex conjunction of constraints on x_2 and x_3. This ability to discover and represent intra-class variance as logical disjunctions is a key advantage of STELLE, allowing it to generalise over the diverse behaviours identified in the local analysis.

Crucially, the high specificity (>93%) and precision (>80%) of these global formulae confirm that whilst the aggregated logic may not capture every instance of a class, the patterns it does identify are highly trustworthy. This indicates that STELLE prioritises correctness over coverage in its global explanations, ensuring that the extracted rules describe valid, high-confidence temporal signatures rather than over-generalised approximations.

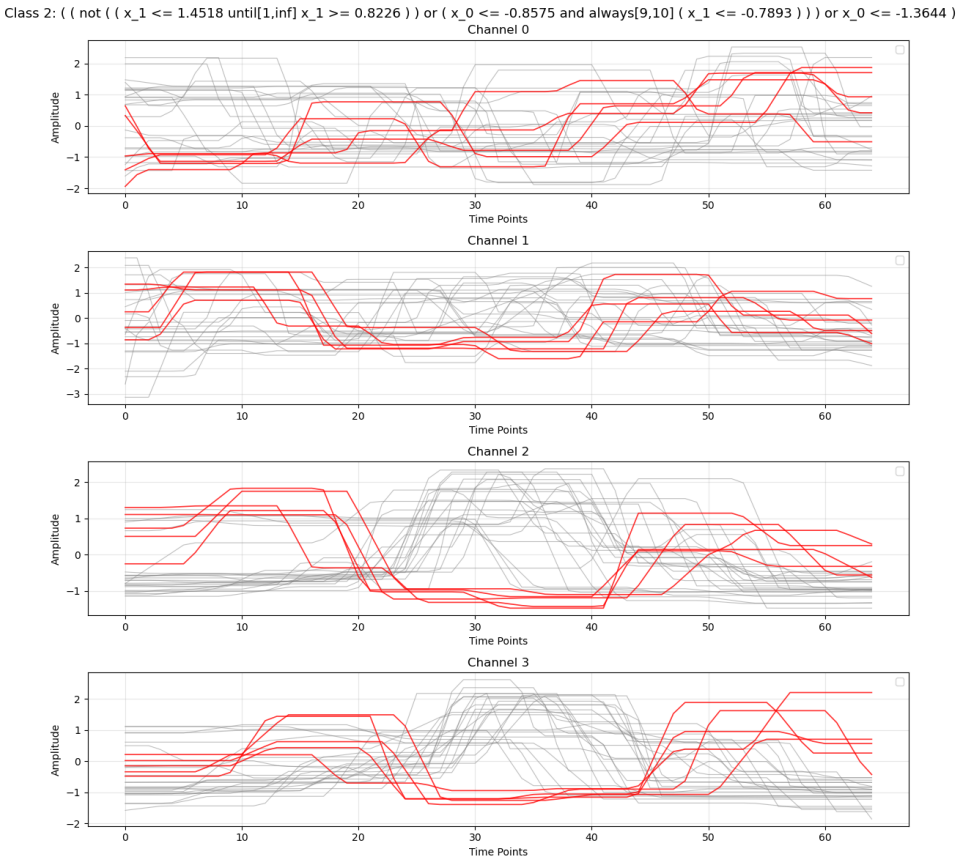


Figure 9.8: Global explanation for Class 2 in the *ERing* dataset. The disjunctive formula captures multiple valid temporal patterns (pathways) that characterise the class, covering the variance observed in the target trajectories (red) relative to the other classes (grey).

Chapter 10

Ablation and Sensitivity Studies

The STELLE framework comprises several interacting components, each designed to contribute to both predictive performance and interpretability. Whilst the complete architecture has been motivated through theoretical considerations and design principles established in previous chapters, it is essential to validate empirically that each component fulfils its intended role and that design choices are well-justified.

This chapter presents a comprehensive suite of ablation and sensitivity studies conducted to evaluate the contribution of individual components, assess the impact of architectural decisions, and examine the robustness of the framework to hyperparameter variations. The studies are organised into four main categories: concept set design (Section 10.3), kernel formulation (Section 10.4), architectural components (Section 10.5), and explanation mechanisms (Section 10.6). All experiments were conducted on a controlled synthetic dataset, enabling precise interpretation of results and isolation of individual effects.

10.1 Experimental Protocol

All ablation and sensitivity studies reported in this chapter were performed on a synthetic dataset specifically designed to exhibit known temporal patterns with controllable characteristics. The dataset construction, statistical properties, and ground-truth temporal structures are detailed in Section 10.1.1. The use of synthetic data offers several advantages for ablation studies: temporal patterns are known a priori, concept ground truth can be established, experimental conditions are fully controlled, and results can be interpreted without confounding factors inherent in real-world data.

For each ablation study, we maintain consistency in training protocol, evaluation metrics, and computational resources, varying only the specific component or hyperparameter under investigation. Unless otherwise stated, all models are trained using the same optimisation procedure, random seeds are fixed for reproducibility, and results are averaged over runs with 3 different seeds to account for stochastic variation. For each test, we explored three learning rates $\eta \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ to ensure that performance differences were not artefacts of suboptimal optimisation.

Following the evaluation protocol established for all ablation studies, we assess the model's performance by to two criteria in order of priority:

1. *Classification accuracy.* Test set accuracy, averaged over three random seeds

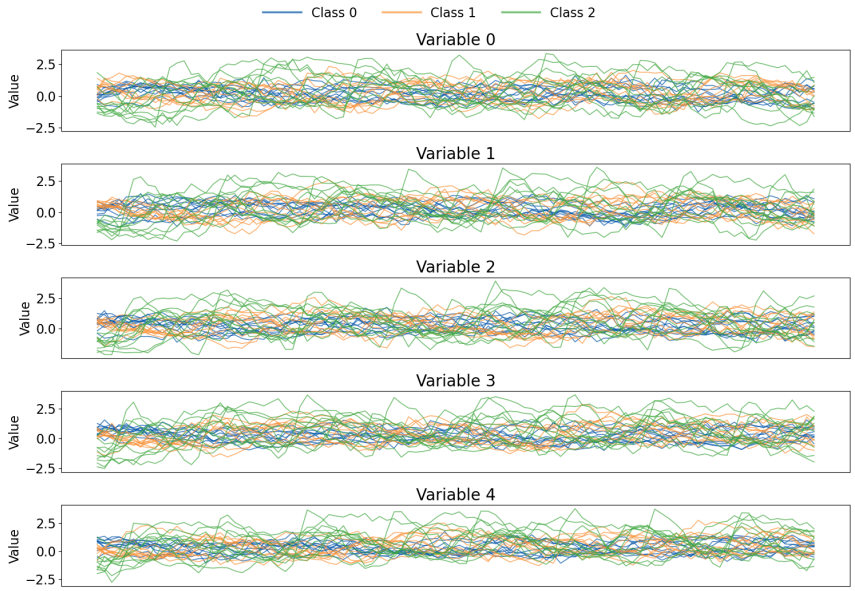


Figure 10.1: Examples of synthetic multivariate time series used for ablation studies (difficulty level 5). Each trajectory contains five variables over 100 time steps across three classes, illustrating class-dependent temporal patterns, noise levels, and trend components.

2. *Explanation conciseness.* We compare the average number of nodes in post-processed local explanations as a measure of interpretability

Only local explanations are evaluated in this study, with cumulative relevance target ratio $\bar{\gamma}_t = 0.8$ and importance parameter $imp_\ell = 0$, to retain maximal explanation separability (Section 8.5.1). Global explanations are not considered here, as they are derived through aggregation of local explanations and therefore inherit the properties of the underlying local explanation mechanism.

10.1.1 Synthetic Dataset

To systematically evaluate model behaviour under controlled conditions, we conduct the ablation studies on a synthetic multivariate time series dataset generated using a custom simulation framework. Examples of the generated trajectories are shown in Figure 10.1. The generator is designed to produce richly parametrised trajectories whose class structure, temporal dynamics, and noise properties can be smoothly adjusted through a difficulty parameter ranging from 1 (easiest) to 10 (hardest). Each trajectory $\mathbf{x} \in \mathbb{R}^{n \times T}$ with n variables and T time points is constructed through a compositional process combining:

- *Periodic components*: class-specific base frequencies with variable-dependent modulation and configurable phase coherence across channels;
- *Harmonic structure*: higher-order frequency components with decaying amplitudes;

- *Class-dependent trends*: linear, quadratic, or exponential drift patterns that vary by class and variable;
- *Non-periodic patterns*: class-specific shape functions (Gaussian, rectified sinusoid, or polynomial) whose contribution is inversely related to periodicity strength;
- *Noise processes*: additive Gaussian noise in both time and frequency domains.

The difficulty parameter controls class separability through ten correlated mechanisms: class separation (reducing inter-class distance), within-class variance (increasing intra-class spread), temporal noise magnitude, frequency-domain noise, periodicity strength (weakening distinctive periodic patterns), frequency variation (reducing consistency within classes), drift strength, outlier probability, phase coherence (reducing cross-variable synchronisation), and trend complexity. These parameters are interpolated linearly between difficulty levels. Complete parameter specifications and generation pseudocode are provided in Appendix B.

For all ablation experiments, we employ the mid-range difficulty setting (difficulty 5), which produces moderately challenging but learnable classification tasks. The synthetic dataset consists of 500 training trajectories and 100 test trajectories, each of length $T = 100$ time steps, with $n = 5$ variables per trajectory and $K = 3$ balanced classes. Training and test sets are generated independently using identical difficulty parameters but different random seeds (seeds 0 and 1 respectively), ensuring that evaluation assesses generalisation rather than memorisation. Class labels are assigned uniformly at random, and class-specific parameters (base frequency, amplitude scaling, phase offset, and shape parameter) are deterministically derived from the class index to ensure reproducibility. This controlled synthetic setup allows us to isolate the contribution of specific architectural components and to analyse the model's robustness across a wide spectrum of temporal behaviours whilst maintaining full knowledge of the ground-truth generative process.

Difficulty calibration. To validate the difficulty parametrisation, we trained four baseline classifiers, ResNet, LSTM, Random Forest, and Logistic Regression, across all ten difficulty levels. Figure 10.2 shows that test accuracy decreases monotonically with increasing difficulty for all methods, confirming that the generative parameters successfully modulate task complexity. At difficulty 1, all deep learning baselines achieve near-perfect accuracy ($\geq 99.6\%$), whilst at difficulty 10, performance degrades to the $87 - 90\%$ range, representing a substantial but not insurmountable classification challenge. The linear classifier exhibits the steepest decline, dropping from $99.8\% to 87.4\%$, indicating that higher difficulty levels introduce non-linear temporal patterns that cannot be captured through simple feature aggregation.

Table 10.1 reports additional diagnostic metrics for selected difficulty levels. Centroid separation decreases from 23.90 (difficulty 1) to 6.31 (difficulty 10), whilst the Davies–Bouldin index increases from 1.33 to 3.77 at difficulty 9, reflecting progressively overlapping class distributions. The anomalous negative Davies–Bouldin value at difficulty 10 (-2.97) indicates extreme cluster degeneracy, suggesting that at the hardest setting, class boundaries become highly irregular and standard clustering metrics may no longer apply. Nevertheless, classifiers still achieve $\sim 90\%$ accuracy at this level,

demonstrating that discriminative patterns remain learnable despite severe geometric overlap.

We selected difficulty 5 as the anchor point for ablation studies because it represents a balanced setting where: (1) classification remains feasible ($92-93\%$ baseline accuracy) but non-trivial, allowing architectural differences to manifest; (2) all baseline methods perform comparably, avoiding ceiling or floor effects; and (3) class separation metrics indicate moderate overlap (centroid separation ≈ 9.80, Davies–Bouldin ≈ 4.30) that requires robust temporal reasoning. This setting ensures that ablation results reflect genuine architectural contributions rather than artefacts of overly simple or excessively difficult data.

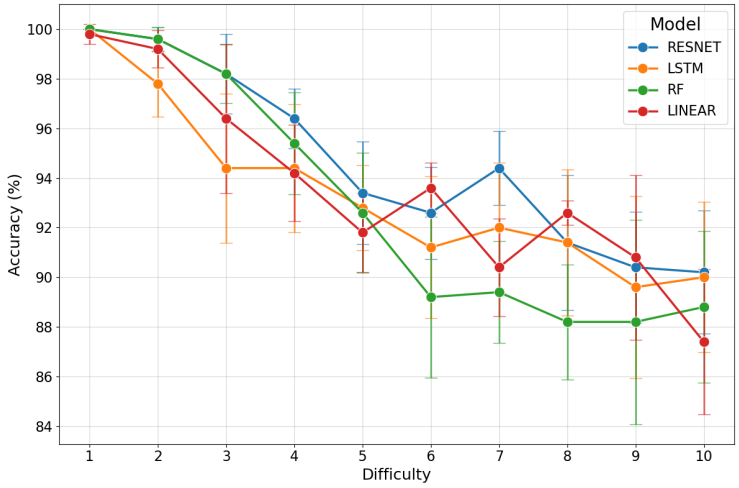


Figure 10.2: Test accuracy of baseline classifiers across difficulty levels. All methods exhibit monotonic performance degradation, confirming successful difficulty calibration. Error bars show standard deviation over five independent seeds. ResNet and LSTM show the most robust behaviour, maintaining $> 90\%$ accuracy even at difficulty 10, whilst Random Forest and Linear classifiers degrade more steeply.

Table 10.1: Diagnostic metrics for difficulty calibration on test set (mean $\pm$ std over five seeds).

Difficulty	ResNet Acc. (%)	LSTM Acc. (%)	Random Forest Acc. (%)	Linear Acc. (%)	Centroid Sep. (ratio)	Davies–Bouldin (index)
1 (easiest)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.8 ± 0.4	23.90 ± 0.78	1.33 ± 0.10
2	99.6 ± 0.5	97.8 ± 1.3	99.6 ± 0.5	99.2 ± 0.7	15.89 ± 0.35	2.52 ± 0.08
3	98.2 ± 1.6	94.4 ± 3.0	98.2 ± 1.2	96.4 ± 3.0	12.05 ± 0.77	3.51 ± 0.23
4	96.4 ± 1.2	94.4 ± 2.6	95.4 ± 2.1	94.2 ± 1.9	9.74 ± 1.11	4.16 ± 0.28
5 (medium)	93.4 ± 2.1	92.8 ± 1.7	92.6 ± 2.4	91.8 ± 1.6	9.80 ± 0.60	4.30 ± 0.26
6	92.6 ± 1.9	91.2 ± 2.9	89.2 ± 3.2	93.6 ± 1.0	10.92 ± 1.09	3.97 ± 0.26
7	94.4 ± 1.5	92.0 ± 2.6	89.4 ± 2.1	90.4 ± 2.0	11.28 ± 1.28	3.65 ± 0.24
8	91.4 ± 2.7	91.4 ± 2.9	88.2 ± 2.3	92.6 ± 0.5	10.55 ± 1.49	3.51 ± 0.26
9	90.4 ± 2.2	89.6 ± 3.7	88.2 ± 4.1	90.8 ± 3.3	9.18 ± 1.64	3.77 ± 0.60
10 (hardest)	90.2 ± 2.5	90.0 ± 3.0	88.8 ± 3.1	87.4 ± 2.9	6.31 ± 2.03	$-2.97 \pm 0.42^*$

* Negative Davies–Bouldin index at difficulty 10 indicates degenerate clustering structure, likely due to extreme overlap between classes.

10.2 Analytical Dimensions of the Study

The analyses presented in this chapter are organised to investigate STELLE along three complementary dimensions: representational capacity, decision mechanism, and explanation structure. While the experimental protocol defines how results are measured, this section clarifies what conceptual aspect of the model each group of experiments is intended to stress.

First, the concept set design (Section 10.3) experiments analyse the representational capacity of the STL embedding space. By varying the number of concepts, their class-specificity, the number of variables per formula, and the similarity threshold, we assess how the structure of the symbolic basis affects both predictive performance and the division metrics. These studies examine whether performance gains arise from richer semantic expressiveness or from redundancy in the concept space.

Second, the kernel formulation (Section 10.4) and architectural components (Section 10.5) analyses evaluate the role of the decision mechanism. Here, we isolate how robustness aggregation, scaling choices, and architectural modules contribute to class discriminability. These experiments test whether the observed performance depends critically on specific modelling decisions or whether the framework remains stable under controlled modifications.

Finally, the explanation mechanisms experiments (Section 10.6) investigate the stability and coherence of the extracted explanations. Since STELLE is designed to balance predictive accuracy with interpretability, it is necessary to evaluate not only classification performance but also the behaviour of global and local division metrics under structural changes.

This structure clarifies the role of each group of experiments and supports a systematic interpretation of their outcomes.

10.3 Concept Set Design

The STL concept set forms the semantic foundation of STELLE's interpretability. The quality, diversity, and size of this concept set directly influence both the expressiveness of the model's internal representations and the computational cost of training and inference. This section examines four key design choices related to concept construction and selection. We begin by analysing how the size of the concept set affects performance and explanations, before comparing two strategies for distributing concepts across variables.

Unless otherwise specified, all experiments in this section use the default STELLE configuration described in Chapter 8, varying only the component under analysis.

10.3.1 Concept Set Size

The size of the concept set $|\Phi_C|$ represents a fundamental trade-off in STELLE's design. Larger concept sets provide greater expressiveness and finer-grained temporal distinctions, potentially improving both classification accuracy and explanation specificity. However, they also increase computational cost during training and

inference, and may introduce redundancy if behavioural diversity does not scale proportionally with set size.

Experimental Setup

We trained STELLE models with concept set sizes $|\Phi_C| \in \{50, 100, 500, 1000, 2000, 3000, 4000, 5000\}$. All other architectural parameters were held constant, including the discriminability mechanism, concept relevance temperature $t = 1.0$, and classifier head architecture. Training proceeded for 2000 epochs with early stopping based on validation accuracy.

Results and Discussion

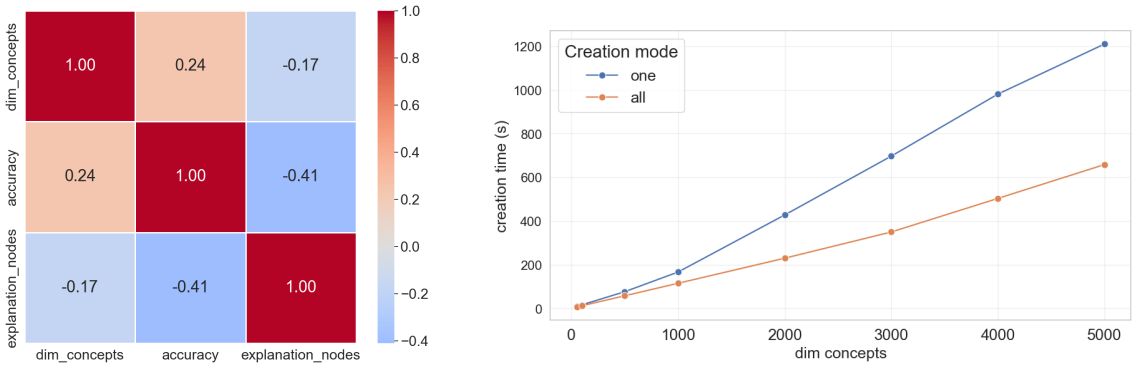
Increasing the size of the concept set has a mild but noticeable effect on performance. accuracy tends to stabilise at slightly higher values as $|\Phi_C|$ grows, and performance becomes more consistent across different learning rate configurations. Smaller concept sets (50–100 concepts) exhibit greater variability: whilst they can occasionally achieve competitive accuracy, they are also more prone to underperformance depending on hyperparameter selection. Larger concept sets ($|\Phi_C| \geq 1000$) produce more stable results, suggesting that a sufficiently diverse concept vocabulary reduces sensitivity to optimiser hyperparameters and makes training more reliable.

However, the improvements are modest: models with 3000–5000 concepts do not significantly outperform those with 500–1000 concepts. Several runs in the mid-range already approach the upper performance bound observed across all configurations, and adding thousands of additional concepts does not consistently yield further gains. This diminishing-return effect is reflected in the weak but positive correlation between concept-set size and accuracy ($r = 0.24$), shown in Figure 10.3a. Concept-set size therefore contributes to performance but is not the dominant factor shaping discriminability.

A more substantial effect emerges in explanation structure and stability. As shown in Figure 10.3a, concept-set size correlates negatively with the number of nodes in the explanations ($r = -0.17$), while accuracy correlates more strongly and negatively with explanation complexity ($r = -0.41$). Together, these trends indicate that larger concept sets indirectly promote simpler explanations by increasing the likelihood of including concepts whose robustness patterns align cleanly with class boundaries.

Concept construction time increases almost linearly with $|\Phi_C|$, as illustrated in Figure 10.3b. The growth is substantial: generating 50 concepts takes on the order of 6–8 seconds, whereas producing 5000 concepts requires between 650 s and 1200 s depending on sampling mode. While these costs are incurred only once, they underscore the practical trade-off between temporal coverage and computational overhead.

Overall, the effect of concept-set size can be summarised as follows: larger concept sets increase temporal coverage and stabilise accuracy, while also improving explanation quality by reducing structural complexity and variability. Yet these benefits plateau beyond a few thousand concepts, at which point additional construction time outweighs the marginal performance and interpretability gains.



(a) (b)

Figure 10.3: (a) Correlation matrix showing the relationships between concept-set size, classification accuracy, and the number of nodes in the explanations. Concept-set size exhibits a weak positive correlation with accuracy and a weak negative correlation with explanation complexity, while accuracy correlates more strongly with simpler explanations. (b) Concept construction time as a function of concept-set size $|\Phi_C|$. Times grow approximately linearly with the number of concepts, reflecting the cost of generating increasingly complex temporal patterns.

10.3.2 Class-Specific vs Unified Concept Sets

When constructing STL formulae for multivariate time series with n variables, two strategies are available. The *unified* approach ("all") generates the full set $|\Phi_C|$ of formulae by randomly sampling the variable indices appearing in each atomic predicate. The *variable-specific* approach ("one") instead constructs $|\Phi_C|/n$ base formulae referencing a single variable (e.g., x_0) and then systematically replicates each formula once for every variable, substituting the corresponding index. Both strategies produce concept sets of the same total size but differ fundamentally in how they distribute concepts across variables.

The variable-specific strategy ensures balanced coverage: each temporal pattern appears exactly once for each variable, which can be advantageous when variables encode qualitatively different behaviours. The unified strategy increases diversity by sampling variables independently for each formula, but this may lead to uneven representation across variables.

Experimental Setup

We compared the two strategies across the full grid of concept-set sizes (50–5000) and learning rates used in the all experiments. All other hyperparameters were kept fixed. We also measured concept construction time for both strategies (Figure 10.3b), as the replication step in the variable-specific approach introduces a different computational pattern compared to fully random generation.

Results and Discussion

The two strategies achieve very similar classification performance across all concept-set sizes and learning rates. accuracy varies mostly with $|\Phi_C|$ and the learning rate, not

with the construction strategy itself. Although the unified strategy obtains a slightly higher average accuracy (86.67% vs.86.46%) this difference is not statistically significant (paired t-test, $p = 0.852$), indicating that both strategies perform equivalently in practice.

Overall, the data shows that the choice of construction strategy has negligible impact on predictive performance. This is consistent with earlier findings that STELLE's discriminative power depends primarily on the robustness-induced semantics induced by the concept set, rather than the specific allocation of variables within each STL formula.

While accuracy is largely unaffected, the two strategies differ more noticeably in explanation structure.
Across runs, formulae originating from the unified strategy tend to require fewer post-processing steps and exhibit slightly lower complexity. In contrast, formulae from the variable-specific strategy show higher variability in readability and, in several runs, higher overall complexity. This effect is not uniform across all concept-set sizes, but the trend is visible. as the variable-specific approach produces more structurally complex explanations on average.

A plausible interpretation is that the unified strategy generates a wider variety of mixed-variable temporal patterns, increasing the likelihood of discovering formulae whose robustness profiles align smoothly with class-level differences. In contrast, the variable-specific method produces sets of tightly related formulae (differing only in variable index), which may over-emphasise variable-specific characteristics and thus require more structural corrections during post-processing.

Concept construction time differs substantially between the two strategies (Figure 10.3b). For all concept-set sizes, the unified approach is consistently faster. The gap widens as $|\Phi_C|$ increases: for 5000 concepts, unified generation requires approximately 660 s, whereas variable-specific replication takes over 1200 s. Thus, while both strategies ultimately produce equally expressive concept vocabularies, the variable-specific method is noticeably more expensive to construct at scale.

10.3.3 Number of Variables per Formula

STL formulae may reference one, two, or more signal variables in their atomic predicates. Univariate concepts ($n_{\text{vars}} = 1$) capture temporal patterns within individual channels, bivariate concepts ($n_{\text{vars}} = 2$) express relationships between pairs of variables, and multivariate concepts ($n_{\text{vars}} \geq 3$) encode higher-order interactions. This subsection examines how restricting or expanding the number of variables allowed in generated concepts affects the discriminative ability of STELLE, the computational cost of concept construction, and the complexity of the resulting explanations. The reported concept generation time includes the robustness-computation stage, which is integral to STELLE's construction pipeline and accounts for most of the computational expense.

Experimental Setup

We generated concept sets of 1000 formulae using the variable-specific method under three configurations, enforcing $n_{\text{vars}} \in \{1, 2, 3\}$ as the maximum number of variables appearing in each formula.

Whilst a fixed concept-set size may appear disadvantageous for higher n_{vars} values, as formulae are distributed across more variable combinations, this choice reflects a practical constraint: in real applications, the concept budget is typically fixed by computational or interpretability requirements. Our goal is therefore to assess how effectively the model exploits different types of temporal relationships under equal representational capacity, rather than to evaluate performance under optimal coverage conditions for each configuration.
For each configuration, we evaluated multiple learning rates and similarity thresholds while keeping all other architectural parameters fixed. For every run, we recorded classification accuracy, concept-generation time, and explanation readability, i.e., the number of nodes in the final explanation, before post-processing.

Results and Discussion

The upper panel of Figure 10.4a reports the relationship between the number of variables per formula and classification accuracy. Across all runs, accuracy remains remarkably stable regardless of n_{vars}. Both univariate and bivariate concepts achieve accuracies predominantly in the 85-90% range, while multivariate concepts ($n_{\text{vars}} = 3$) occasionally produce isolated peaks at 95% and, in a single configuration, 100%. These peaks, however, do not represent a consistent improvement, as most multivariate runs fall back within the same accuracy band as the lower-dimensional settings. This indicates that, for the present dataset, temporal shape characteristics are sufficiently discriminative even without modelling higher-order cross-channel interactions.

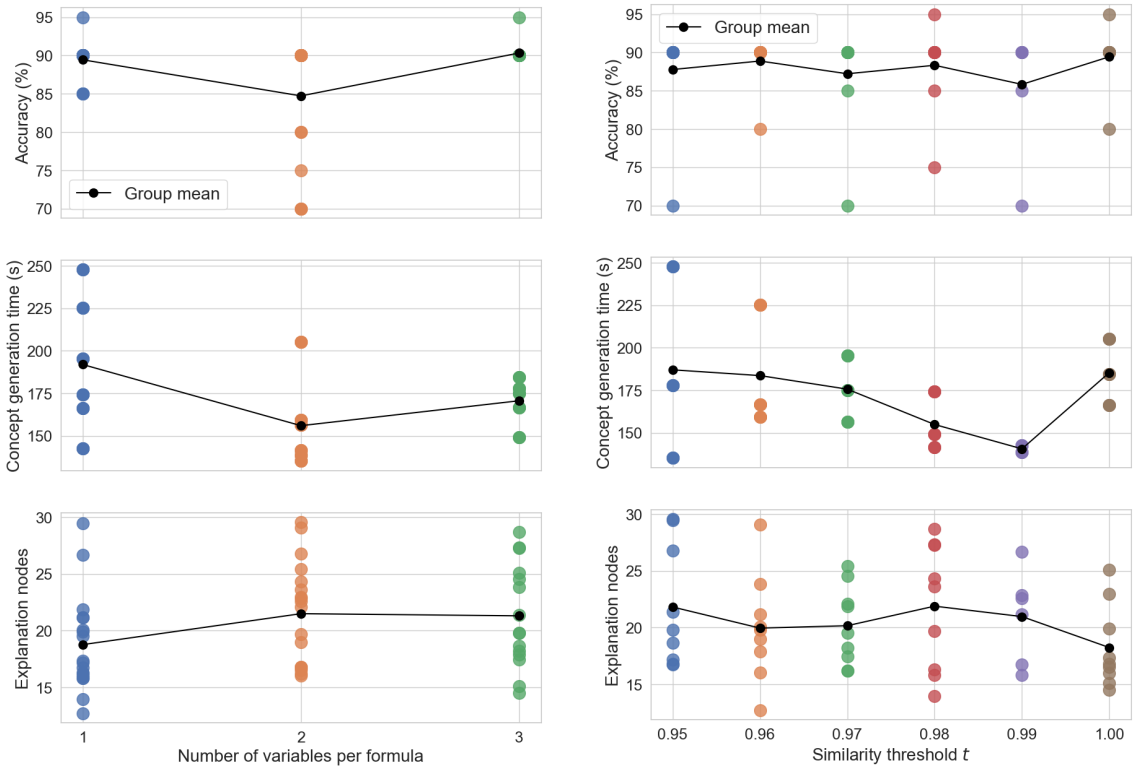
The middle panel of Figure 10.4a shows that the computational cost of concept generation does not increase with the number of variables per formula. In fact, the $n_{\text{vars}} = 2$ configuration systematically yields the lowest generation times, while the univariate and multivariate settings are slightly slower but broadly comparable. This suggests that allowing additional variables in atomic predicates does not introduce a meaningful overhead during concept construction.

Lastly, the bottom panel of Figure 10.4a summarises explanation readability before post-processing. Readability exhibits high variability across all three settings, with minimum values around 12-16 nodes and occasional increases to 20-30. Crucially, no monotonic relationship emerges between n_{vars} and formula complexity. Increasing the number of variables does not consistently simplify explanations, nor does it systematically make them harder to read. These results indicate that explanation complexity is driven more by the robustness landscape of individual formulae than by the dimensionality of the underlying predicate space.

These results further support the conclusion that temporal structure, rather than cross-variable interaction, is the primary source of discriminative signal for this classification task.

10.3.4 Effect of the Similarity Threshold t

The parameter t regulates similarity-based filtering during concept selection. Values below one impose a minimum dissimilarity between retained formulae, promoting diversity and reducing redundancy, whereas $t = 1$ disables the filter and allows the full set of generated formulae to be used. This subsection examines how varying t



(a) (b)

Figure 10.4: Ablation results on concept construction design choices. (a) Effect of restricting STL concepts to use 1, 2, or 3 variables. For each value of n_{vars}, we report accuracy, concept-generation time, and the size of the explanations. (b) Effect of the similarity threshold t used for diversity filtering during concept selection. Lower values of t enforce stronger dissimilarity and reduce redundancy, while $t = 1$ disables filtering. Each panel shows individual runs as scatter points and the mean trend as a solid line.

affects the size and structure of the concept vocabulary, and in turn the classification accuracy, the computational cost of concept construction, and the complexity of the resulting explanations. The reported concept generation time includes the robustness-computation stage, which is integral to STELLE's construction pipeline and accounts for most of the computational expense.

Experimental Setup

We evaluated STELLE across similarity thresholds $t \in \{1.00, 0.99, 0.98, 0.97, 0.96, 0.95\}$ while also varying the learning rate and the number of variables per formula. All other components of the architecture were kept constant, including the number of concepts, fixed at 1000, created by using the variable-specific method. For each configuration we measured predictive performance, concept-generation time, and the number of nodes in the raw explanation, before being postprocessed.

Results and Discussion

The effect of the similarity threshold on accuracy is summarised in the top panel of Figure 10.4b. Across all configurations, accuracy remains remarkably stable. No value of t yields a consistent advantage, and disabling the filter ($t = 1$) does not reduce performance. These results indicate that the discriminative temporal patterns captured by the concept set are robust to moderate levels of redundancy.

The middle panel of Figure 10.4b shows a clear and monotonic relationship between the similarity threshold and concept-generation time. Lower thresholds ($t < 1$) consistently lead to longer runtimes. This occurs because stricter dissimilarity constraints cause a larger portion of candidate formulae to be rejected as too similar, forcing the construction procedure to explore additional candidates until the target number of concepts is reached. As a result, concept generation becomes slower the more aggressively similarity filtering is enforced.
The case $t = 1$ behaves differently. Here, similarity filtering is disabled, so every sampled formula is accepted without comparison. However, when $t < 1$, robustness values must be computed incrementally during formula generation to enable similarity-based filtering, and these values are cached for subsequent use. In contrast, when $t = 1$, no intermediate robustness computation is required, but all robustness values must be computed in a single batch at the end of the generation process. This deferred batch computation introduces a fixed overhead that prevents $t = 1$ from achieving the lowest runtime, despite the absence of filtering operations. Consequently, the fastest generation times occur at moderate thresholds ($t \approx 0.7$–0.9), where filtering overhead remains low but robustness caching still provides computational benefits.

Finally, the bottom panel of Figure 10.4b displays explanation readability. Readability values vary within a similar range across all thresholds, with no clear trend indicating that stronger filtering leads to simpler or more complex explanations.

Taken together, these findings show that the similarity threshold t has only a marginal effect on classification accuracy, computational efficiency, and explanation complexity. While the threshold does influence the internal composition of the concept set, its downstream impact on STELLE's behaviour is limited. This provides further evidence of the robustness of the model with respect to the structure of the concept vocabulary.

10.3.5 Summary of Findings

The experiments on concept set design show that the structure and composition of the STL basis primarily influence interpretability metrics, while predictive accuracy remains comparatively stable across moderate configurations. Increasing the number of concepts improves global and local division up to a saturation point, beyond which gains become marginal. Class-specific concept sets enhance discriminability but may introduce redundancy, whereas unified sets favour compactness at the cost of slightly reduced separation. The number of variables per formula and the similarity threshold further regulate the trade-off between semantic expressiveness and compactness. Overall, performance improvements arise from meaningful semantic coverage rather than from sheer expansion of the concept space.

10.4 Kernel Formulation

The kernel framework of STELLE consists of two components, both introduced in Chapter 8.3: a formula–formula kernel, used exclusively during concept generation to ensure diversity among candidate STL formulae, and trajectory–formula kernel, which is used to embed trajectories into the concept space and forms the actual input to the classifier. Both kernels admit lightweight transformations, such as normalisation and exponentiation, that influence the scale and distribution of similarity values. Although these operations do not change the logical structure of STL formulae or the post-processing mechanism, they may affect both which concepts are selected and how effectively the model can learn from them. This section evaluates these effects.

Unless otherwise specified, all experiments in this section use the default STELLE configuration described in Chapter 8, varying only the component under analysis.

10.4.1 Normalisation and Exponentiation

Normalisation and exponentiation play different functional roles depending on whether they are applied to the formula–formula kernel or to the trajectory–formula kernel. The parameters suffixed with "_kernel" act on the formula–formula similarities used during concept selection: they indirectly shape the concept vocabulary by changing which formulae are judged too similar and therefore filtered out. The parameters suffixed with "_rhotau" instead act directly on the trajectory–formula similarities that define the embedding of each trajectory. These transformations influence the geometry of the embedded space, the magnitude of gradients during optimisation, and ultimately the model's discriminative performance. Since the structure of the explanations depends only on the selected concept set and the post-processing algorithm, and not on the numerical scaling of ρ_τ values, we focus our analysis on predictive accuracy rather than readability.

Experimental Setup

We evaluate all combinations of four binary operations: `normalize_kernel`, `exp_kernel`, `normalize_rhotau`, and `exp_rhotau`. For each configuration, we train STELLE with three random seeds and three learning rates, keeping all other parameters fixed. The concept set is regenerated whenever kernel-side parameters are modified, ensuring that their influence on concept diversity is correctly captured. Trajectory–formula transformations (`_rhotau`) are applied after concept generation and therefore influence only trajectory embeddings.

Results and Discussion

Figure 10.5 reports the main effects of the four transformations on average accuracy. Two consistent trends emerge.

First, normalisation is broadly beneficial on both kernels. Normalising the formula–formula kernel stabilises similarity values during diversity filtering, making the selection process less sensitive to outliers and yielding more consistent concept sets. Normalising the trajectory–formula kernel has an even clearer positive effect.

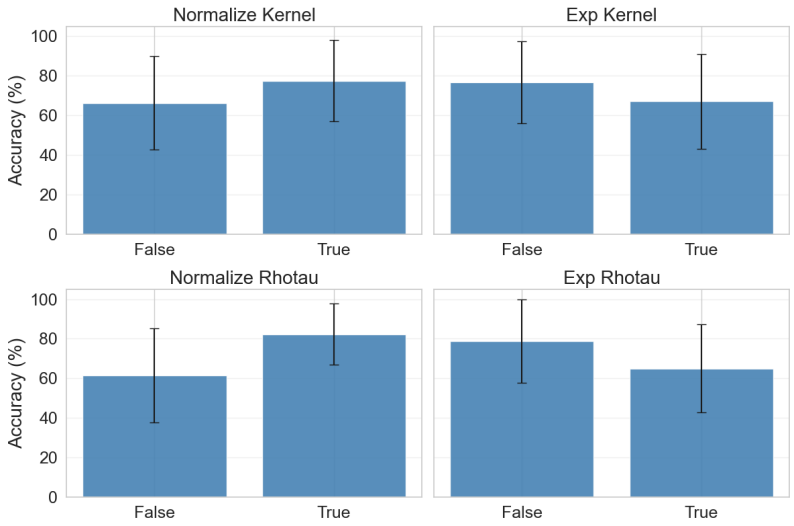


Figure 10.5: Main effects of normalisation and exponentiation on accuracy. Each panel shows the mean accuracy obtained when toggling one of the four scaling operations. Error bars denote one standard deviation over seeds and learning rates. Normalisation consistently improves performance; exponentiation is beneficial only on the formula–formula kernel and detrimental on the trajectory–formula kernel.

Although the ρ_τ values are not fed directly to the classifier, they form the basis of the trajectory embedding from which relevance weights, prototype distances, and ultimately class scores are computed. When these values lie within a stable numerical range, all downstream computations such as concept relevance calculations, distance evaluations, and gradient updates, become more stable. Across seeds and learning rates, configurations using normalised ρ_τ values achieve consistently higher accuracy.

Second, exponentiation has asymmetric effects across different kernel components. When examining the main effects in isolation (Figure 10.5), exponentiating either the formula–formula kernel (`exp_kernel`) or the trajectory–formula kernel (`exp_rhotau`) does not consistently improve performance. However, the interaction analysis (Figure 10.6) reveals that the optimal configuration combines normalisation and exponentiation for the formula–formula kernel with normalisation but *no* exponentiation for the trajectory–formula kernel (`norm=1`, `exp=0` for both kernels, `norm=1`, `exp=1` for formula–formula only).

This suggests that exponentiation plays different roles depending on where it is applied. Exponentiating the formula–formula kernel (`exp_kernel`) selectively increases contrast among moderately similar formulae, producing concept sets with more diverse temporal patterns when combined with normalisation. In contrast, exponentiating ρ_τ values appears to introduce unnecessary amplification of small trajectory differences, potentially creating a noisier embedding space without corresponding gains in discriminative power.

The apparent contradiction between the marginal effects (which show no clear benefit to exponentiation) and the best configuration (which uses exponentiation for the formula kernel) highlights the importance of interaction effects in kernel design. Whilst the isolated impact of exponentiation is modest, its contribution within the right architectural context proves beneficial. We therefore adopt the configuration `norm=1`,

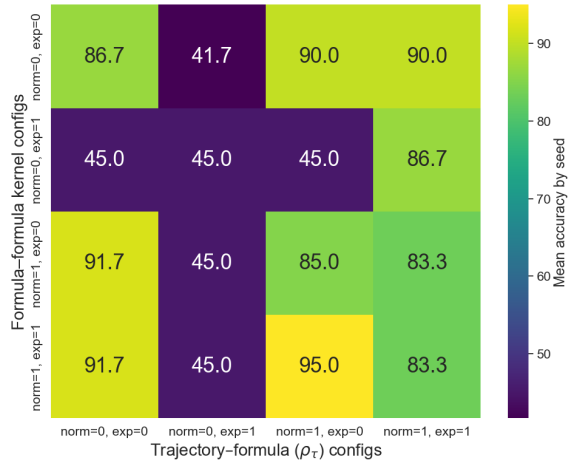


Figure 10.6: Mean accuracy across all 16 combinations of kernel and trajectory–formula transformations. The optimal configuration (normalised and exponentiated formula–formula kernel, normalised but non-exponentiated trajectory–formula values) clearly appears as the global maximum.

`exp_kernel=1`, `exp_rhotau=0` for subsequent experiments, reflecting the interplay between normalisation and selective exponentiation across different kernel components.

10.4.2 Kernel Regression as a Diagnostic Ablation

Whilst other experiments evaluate STELLE in terms of classification accuracy and explanation quality, they do not directly assess the adequacy of the underlying trajectory–formula kernel on which the framework is built. Since STELLE relies on this kernel to embed trajectories and STL formulae into a shared semantic space, its performance is inherently constrained by how well the kernel captures meaningful relationships between temporal logic concepts and data.

This ablation study investigates whether datasets on which STELLE performs less favourably correspond to regimes where the kernel itself provides a poor approximation of formula semantics. The core hypothesis is that reduced classification accuracy is not necessarily due to limitations of the classifier or explanation pipeline, but may instead reflect a mismatch between the chosen kernel (and associated reference measure μ_0) and the structure of the data.

To probe this hypothesis, we frame a set of auxiliary kernel regression tasks defined directly at the level of STL formulae. In these tasks, the kernel is used to predict the behaviour of previously unseen formulae based on their similarity to known formulae, evaluated under different targets such as average robustness across the dataset, or Boolean satisfiability with respect to the training signals. These tasks provide a direct diagnostic of how well the kernel preserves semantic relationships between formulae and data.

By comparing kernel regression performance across datasets and analysing its correlation with STELLE's classification accuracy, this study aims to identify whether poor predictive performance arises in settings where the kernel is intrinsically ill-suited. A strong correlation would indicate that STELLE degrades precisely when

the kernel fails to model formula behaviour reliably, suggesting that such failures are expected given the kernel choice. This analysis further motivates the exploration of alternative kernel formulations or dataset-adaptive choices of μ_0 as potential directions for improvement.

Experimental Setup

We consider a set of datasets spanning both high and low STELLE classification accuracy, namely those introduced in Chapter 9.1. A set of STL formulae is generated using the same procedure and parameters adopted throughout the thesis, and temporally rescaled to match the length of the corresponding signals. Kernel regression is then performed at the level of STL formulae using the same trajectory–formula kernel employed by STELLE. For each dataset, the available formulae are split into a training set and a held-out test set. The kernel is used to predict the behaviour of each test formula as a weighted combination of training formulae, where weights are obtained by solving a regularised kernel regression problem.

We evaluate multiple regression targets, namely average robustness across all trajectories in the dataset and Boolean satisfiability with respect to the training signals. Regression performance is quantified using mean squared error and R^2, and analysed in relation to STELLE's classification accuracy on the same dataset. In addition, the distribution of prediction errors is examined to characterise systematic kernel failure modes.

Results and Discussion

The quantitative results of the kernel regression ablation are summarised in Table 10.2. We report the coefficient of correlation (R^2) and Mean Squared Error (MSE) for the prediction of average formula robustness (ρ) and satisfiability ($\models$). To facilitate the analysis of correlations, the datasets are grouped by their associated STELLE classification accuracy.

Target Complexity: Robustness vs. Satisfiability. A definitive trend observed across all datasets is the disparity in performance between predicting continuous robustness values and binary satisfiability. As shown in Table 10.2, the regression R^2 for satisfiability is universally negative, indicating that the kernel fails to capture the sharp decision boundaries associated with Boolean truth. Conversely, the prediction of average robustness yields significantly higher R^2 scores (e.g., 1.00 for *Maritime*, 0.97 for *BasicMotions*). This confirms that the trajectory–formula kernel is inherently suited to the smooth landscape of quantitative semantics (robustness) rather than the discrete landscape of logic. Consequently, the remainder of this discussion focuses on the robustness regression metric as the primary diagnostic tool.

Correlation with Classification Performance. The results largely support the hypothesis that the kernel's ability to model formula semantics is linked to classification success. For datasets where STELLE achieves near-perfect accuracy, such as *Maritime*, *GunPointOldVersusYoung*, and *BasicMotions*, the kernel regression task is also solved with high precision ($R^2 \geq 0.96$). This suggests that in these domains, the kernel

successfully maps syntactically similar formulae to semantically similar behaviours, creating a coherent embedding space that facilitates both regression and classification.

Conversely, datasets with lower classification accuracy often exhibit poor regression performance. For instance, *AtrialFibrillation* (accuracy 0.36) and *Handwriting* (accuracy 0.20) yield negative R^2 values (-4.96 and -0.01, respectively). In these cases, the kernel fails to generalise: the behaviour of unseen formulae cannot be reliably predicted from training examples. This implies that the kernel metric μ_0 may be ill-suited to the signal topology of these specific datasets, leading to a "semantic mismatch" that hampers the entire pipeline.

Regime of Semantic Separability. An interesting deviation is observed in the *EthanolConcentration* dataset. Here, the kernel regression performs almost perfectly ($R^2 \approx 1.00$, MSE ≈ 0), yet STELLE's accuracy remains low (0.32). This indicates a regime where the kernel is mathematically well-behaved, successfully capturing the relationship between formula structure and signal output, but the classes themselves may not be distinguishable via temporal logic features. In other words, a good kernel is a necessary condition for success, but it is not sufficient if the underlying physical phenomenon lacks discriminative temporal patterns.

Outliers and Numerical Sensitivity. Finally, we note specific cases such as *Epilepsy2* and *NerveDamage*, where high classification accuracy coexists with extremely poor regression scores ($R^2 \ll -1$). This likely points to a difference in task difficulty: the classifier (SVM) only requires the data to be linearly separable in the kernel space, whereas regression requires the kernel to support smooth interpolation across the entire manifold. The failure of regression here suggests that while the kernel space is fragmented or non-smooth (making prediction hard), the classes remain sufficiently disjoint to allow for accurate categorisation.

10.4.3 Summary of Findings

The kernel formulation plays a central role in preserving class discriminability within the STL embedding space. Variants that weaken or simplify robustness aggregation consistently reduce division metrics and, in some cases, predictive accuracy. These results indicate that the quantitative robustness semantics are not merely a feature transformation, but a structural component enabling separation in the joint embedding space. The stability of performance under moderate kernel adjustments confirms robustness of the framework, while degradations under simplified formulations highlight the importance of preserving semantic alignment between trajectories and formulae.

10.5 Architectural Components

To understand which components of STELLE's architecture are responsible for its predictive power and interpretability, we evaluate several ablated variants obtained by selectively disabling or modifying key elements of the embedding pipeline. These variants isolate the contribution of trajectory embedding, robustness-based

Table 10.2: Kernel regression performance across representative UCR datasets. The table compares STELLE classification accuracy against the kernel's ability to predict formula properties (R^2 score). Datasets are grouped by high, mixed, and low classification performance.

Dataset	STELLE Acc.	Robustness R^2	Robustness MSE	Satisfiability R^2
High accuracy Regime				
Maritime	1.00	**1.00**	0.00	-1.00
GunPointOldVersusYoung	1.00	**1.00**	0.00	-4.00
BasicMotions	0.99	0.97	0.00	-2.98
GunPoint	0.92	0.39	0.20	-1.86
Mixed / Anomalous Regime				
Epilepsy2	0.95	-4.3×10^5	3.6×10^4	-4.1×10^5
NerveDamage	1.00	-35.01	1.66	-85.97
EthanolConcentration	0.32	**1.00**	0.00	-4.04
Low accuracy Regime				
Libras	0.75	-9.86	1.05	-22.75
SharePriceIncrease	0.62	0.75	0.13	-2.71
AtrialFibrillation	0.36	-4.96	0.33	-9.14
Handwriting	0.20	-0.01	0.03	-7.08

discriminability mechanism, and hybrid interaction terms. The comparison focuses on four dimensions: mean accuracy, training time, explanation complexity, and test-time cost. The results are summarised in Figure 10.7.

All experiments in this section use the default STELLE configuration described in Chapter 8, varying only the component under analysis, unless otherwise specified.

Experimental Setup

All architectural variants were trained under identical conditions, using the same STL concept set, hyperparameter search space, and evaluation protocol adopted in the main experiments. The ablations isolate the contribution of each structural component of the STELLE architecture by selectively removing, replacing, or restructuring the trajectory embedding or discriminability streams. The following variants were considered:

- *Anchor.* In this setting, trajectories are not embedded directly into the STL concept set through the trajectory–formula kernel. Instead, both trajectories and STL concepts are embedded into a shared *anchor set* consisting of predetermined reference formulae. Each input trajectory is represented through its similarities to the anchors, computed via $\rho_{\tau_j}(x)$, and each STL concept is mapped to the same space by evaluating the formula–formula kernel (introduced in Section 8.3). The final embedding of a trajectory into the concept space is obtained indirectly by composing these two anchor-based representations. Theoretically, such an anchor factorisation could provide a more stable and globally consistent behavioural coordinate system, acting as a fixed basis that captures representative temporal patterns across the dataset. This may, in principle, reduce sensitivity to optimisation noise and produce smoother embeddings. However, the lack of adaptivity and the doubled computational overhead limit its practical effectiveness.

- *No* $G(\mathbf{x})$. Here, the discriminability mechanism is removed entirely. In this variant, the model no longer evaluates class-specific atypicality of concept activations, relying solely on robustness-based embeddings $H(x)$ to discriminate among classes. Theoretically, this could be advantageous if robustness values were already well-separated across classes, since eliminating $G(x)$ reduces computational cost and removes a potential source of noise amplification. In practice, however, the embedding alone is insufficiently informative for challenging datasets.

- *Robustness only.* The classifier operates directly on raw robustness values $\rho(\varphi_i, x)$, effectively turning STELLE into a purely symbolic robustness-based classifier. This setting tests the hypothesis that the temporal concepts themselves, without any geometric or statistical refinement, may already encode separability. Such a formulation is theoretically appealing due to its simplicity and complete semantic transparency, but it often suffers from noisy robustness landscapes that hinder reliable discrimination.

- *Robustness as* $G(\mathbf{x})$. In this variant the discriminability mechanism is replaced with robustness values while the trajectory embedding $H(\mathbf{x})$ is preserved. This assesses whether the kernel-based representation of trajectories suffices to distinguish classes, even when class-specific atypicality is not explicitly modelled, but some information on raw robustness is still exploited. This variant is theoretically motivated by the possibility that the stream of $H(\mathbf{x})$ may dominate discrimination, allowing the other to be simplified.

- *Robustness as* $H(\mathbf{x})$. The trajectory embedding layer is replaced by direct robustness values while the discriminability mechanism $G(\mathbf{x})$ remains intact. This explores whether the discriminability mechanism alone can compensate for the absence of the trajectory–formula kernel by identifying atypical concept activations even when the embedding is removed. Again, this variant theoretically motivated by the possibility that one stream may dominate discrimination.

- STELLE. The complete architecture, combining trajectory embeddings derived from the trajectory–formula kernel and the discriminability mechanism that highlights atypical temporal patterns. This configuration preserves the full interplay between semantic grounding, class-dependent deviation analysis, and structural refinement, and therefore serves as the reference against which all ablated models are compared.

For every configuration, explanation complexity is measured before post-processing, capturing the intrinsic syntactic size of the extracted formulae before any logical simplification or threshold adjustment takes place.

Results and Discussion

The architectural ablations reveal clear differences in predictive performance, computational behaviour, and explanation complexity, with STELLE emerging as the most reliable and balanced configuration (Figure 10.7).

In terms of accuracy, STELLE outperforms all other variants, reaching 93.33%, with corresponding gains in sensitivity and specificity. The next best-performing variants,

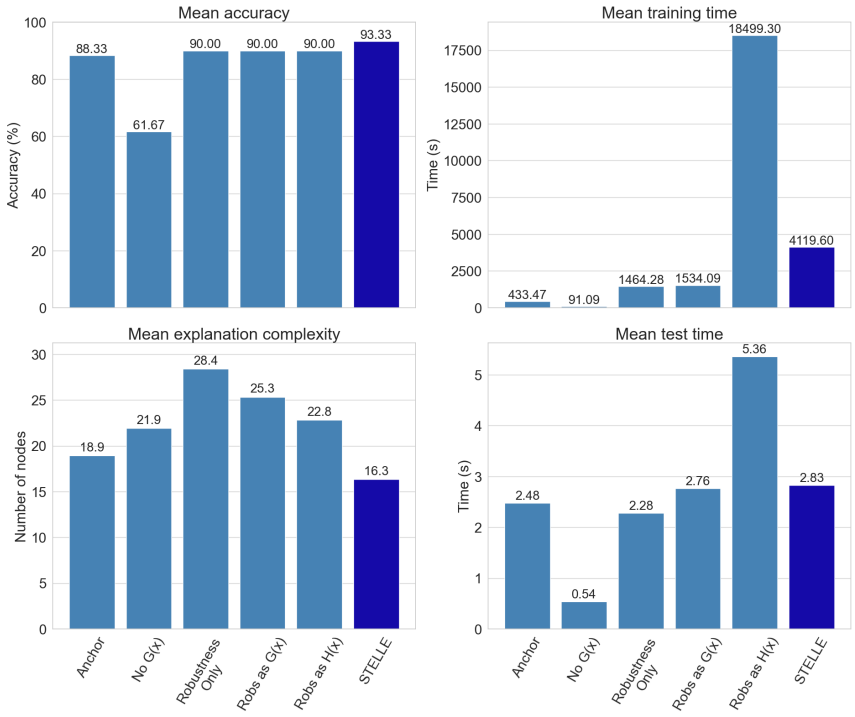


Figure 10.7: Comparison of architectural variants of STELLE. The four panels show mean accuracy, mean training time, mean explanation complexity (number of nodes in the formula before post-processing), and mean test-time cost, computed across seeds. STELLE achieves the best overall trade-off, combining the highest accuracy with efficient training and the lowest explanation complexity. Robustness is abbreviated as "Robs".

Robustness Only, *Robs as trajectory embedding* $H(\mathbf{x})$, and *Robs as discriminability mechanism* $G(\mathbf{x})$, all achieve 90% accuracy. This confirms that robustness values alone contain meaningful discriminative information, but the full hybrid design of STELLE provides a richer representation. The *Anchor* and *No* $G(\mathbf{x})$ variants further illustrate the importance of the learned embedding streams: fixing the prototype-based structure reduces accuracy to 88.33%, while removing the discriminability mechanism $G(\mathbf{x})$ altogether causes accuracy to collapse to 61.67%.

Training time behaviour also varies substantially, largely driven by differences in convergence rates under our validation-based early stopping protocol. The *Robs as* $H(\mathbf{x})$ variant was exceptionally costly (nearly 18,500s), indicating that raw robustness values induce a rough optimisation landscape that forces the model to train for vastly more epochs to stabilise. In contrast, STELLE converges rapidly ($\sim$500s), demonstrating that the kernel-based embedding provides a smoother geometry that facilitates efficient learning compared to the raw robustness baselines (approx. 1,500s).

STELLE itself incurs a higher training time (around 4,100 seconds) compared to most ablated variants, a natural consequence of integrating *both* components: trajectory embedding $H(\mathbf{x})$ and discriminability mechanism $G(\mathbf{x})$. Crucially, this additional computation leads to clear gains in accuracy and a pronounced reduction in explanation complexity.

The *Anchor* architecture offers an instructive contrast. Its training time appears relatively low, but its end-to-end runtime is substantially higher once concept generation is included. Because the Anchor variant requires computing *two* robustness sets, one for the standard concept set and one for the anchor set, the preprocessing cost is effectively doubled. This overhead does not appear in the training time plot, which measures only the learning phase, but it materially affects the practical runtime and reduces the model's appeal despite its moderate accuracy.

Explanation complexity, measured before post-processing, also separates the architectural choices. STELLE produces the simplest explanations (mean number of nodes 16.3), demonstrating that its hybrid structure enables concise discriminative patterns without redundant temporal constraints. In contrast, the *Robustness Only* variant yields the most complex explanations (mean 28.4), supporting the intuition that raw robustness values are noisier and force the model to assemble larger formulae to achieve separation. The robustness-replacement hybrids similarly generate more complex explanations, while the *Anchor* variant produces moderately compact but still less concise formulae relative to STELLE.

Test time behaviour follows similar trends. The *No* $G(\mathbf{x})$ model is the fastest at inference, but its accuracy is too low for practical use. STELLE maintains efficient inference (around 2.8 seconds on average), comparable to or faster than variants with similar accuracy. The slowest configuration is again *Robs as* $H(\mathbf{x})$, reflecting the instability and computational overhead already seen during training.

Taken together, these ablations confirm that the interplay between the trajectory embedding $H(\mathbf{x})$, the discriminability mechanism $G(\mathbf{x})$, and robustness semantics is essential. STELLE offers the best combination of accuracy, explanation simplicity, and test time efficiency. Removing or altering any of these components either diminishes discriminative power, increases training or inference costs, or leads to substantially more complex symbolic explanations. These results demonstrate that the full STELLE

architecture is not only the most effective but also the most computationally balanced design among all tested variants.

10.5.1 Summary of Findings

The ablation of architectural components reveals that STELLE's performance does not rely on architectural complexity alone, but on the interaction between semantic embedding and discriminative aggregation. Removing or modifying individual modules leads to measurable reductions in division metrics, even when accuracy remains relatively stable. This suggests that certain components contribute primarily to explanation coherence rather than to raw predictive performance. The results support the view that interpretability in STELLE emerges from structural integration of robustness information rather than from post-hoc explanation layers.

10.6 Explanation Mechanisms

Whilst the classifier's architecture determines predictive accuracy, the quality of STELLE's explanations depends on several design choices in the explanation pipeline. The ablation studies in this section evaluate each component independently. Unlike the other ablations, which examine classification performance, these analyses focus exclusively on explanation quality. Crucially, all experiments use a fixed, pre-trained classifier, ensuring that observed differences arise solely from the explanation mechanism rather than from changes in predictive behaviour.

We investigate two aspects of the explanation pipeline: the effect of the aggregation strategy on the distinctiveness of extracted concepts, and the influence of hyperparameters such as $\bar{\gamma}_t$ and imp_ℓ on the balance between readability and coverage. The ablation studies focus on how these design choices shape the structure, compactness, and discriminative power of the resulting explanations. The findings provide insights into which components are critical for maintaining explanation quality and which hyperparameter regimes yield stable, compact, and discriminative temporal statements.

All experiments in this section use the default STELLE configuration described in Chapter 8, varying only the component under analysis.

10.6.1 Aggregation Strategies

This ablation study investigates the role of the aggregation strategy used to compute the instance-level discriminative score $\mathbf{r}(\mathbf{x})$ for explanation extraction. The aggregation operator determines how activations associated with the target class are contrasted against those of competing classes, and therefore directly influences which concepts are deemed most distinctive. The goal of this analysis is to assess how different aggregation choices affect explanation quality, independently of the underlying predictive model.

Experimental Setup

We consider three aggregation strategies for computing $\mathbf{r}(\mathbf{x})$. The first corresponds to the original formulation introduced in Equation (8.7), in which the activation of class

k is contrasted against the mean activation across all alternative classes. Two variants are evaluated for comparison: a `max` strategy, where the class-k activation is contrasted against the maximum activation among competing classes, and a `none` strategy, which relies directly on the raw class-k activation without explicit comparison. All other components of the architecture and training procedure are kept fixed. Explanation quality is assessed using the readability score computed as the average number of nodes before post-processing.

Results and Discussion

The results indicate that the `mean` and `max` aggregation strategies produce very similar readability scores, suggesting that both approaches lead to explanations of comparable compactness. In contrast, the `none` variant consistently yields higher readability values, corresponding to less concise explanations. This highlights the importance of explicitly contrasting class-k evidence against alternative classes when extracting discriminative explanations.

Although the `mean` and `max` strategies behave similarly in practice, the `mean` aggregation is theoretically more consistent with the intended notion of concept distinctiveness. By subtracting the average activation over all competing classes, the discriminative score captures how strongly a concept separates the target class from the entire decision context, rather than from a single competing class. This formulation mitigates sensitivity to isolated or outlier activations and provides a more stable estimate of distinctiveness, particularly in multiclass settings. For this reason, and given its empirical equivalence to the `max` variant, the `mean` aggregation is adopted as the default strategy for explanation extraction in the remainder of this work.

10.6.2 Effect of Local Explanation Parameters

This ablation study investigates the effect of the cumulative relevance threshold $\bar{\gamma}_t$ and the importance parameter imp_ℓ on the quality of the extracted local explanations. These two hyperparameters operate at distinct stages of the explanation pipeline and serve complementary roles. The threshold $\bar{\gamma}_t$ is applied during the initial extraction phase, where candidate STL formulae are ranked according to their relevance scores and a preliminary subset of high-scoring concepts is retained. At this stage, selection is purely ranking-based and no post-processing or coverage constraints are enforced, such that $\bar{\gamma}_t$ solely controls how much of the relevance mass is considered for explanation construction. Subsequently, candidate formulae are processed in descending order of relevance and iteratively incorporated into the explanation set. During this post-processing phase, the importance parameter imp_ℓ enforces a minimum level of local separability by specifying how many data points the explanation must correctly divide. If this requirement is not met, additional formulae from the relevance ranking are included until the constraint is satisfied. As a result, $\bar{\gamma}_t$ governs relevance-based pruning during candidate extraction, while imp_ℓ acts as a coverage-based stopping criterion during explanation refinement. The aim of this analysis is to assess how sensitive local separability and explanation readability, measured in terms of node count, are to these hyperparameters and to identify regimes that yield stable and compact explanations.

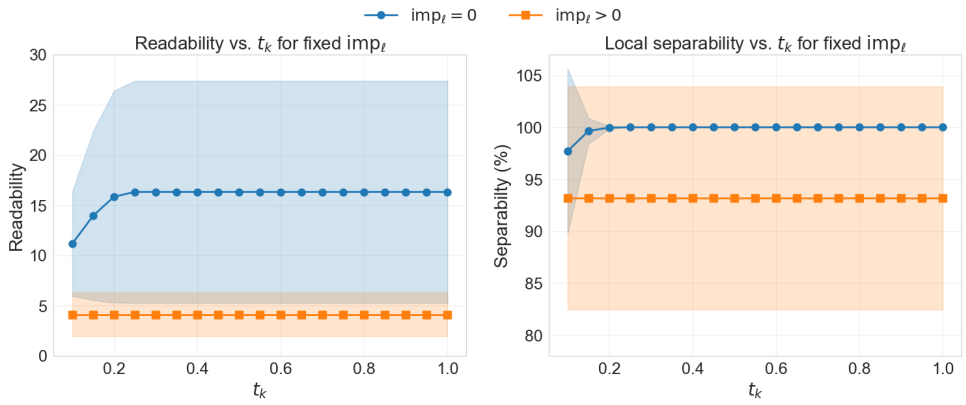


Figure 10.8: Effect of the relevance threshold t_k and the importance parameter imp_ℓ on explanation quality. (left) Readability, measured as the mean number of nodes in the extracted local explanations (shaded areas denote standard deviation). When $imp_\ell = 0$, explanations are longer and less stable, while enforcing $imp_\ell > 0$ yields consistently more compact and stable explanations, independently of t_k. (right) Local separability, measured as the percentage of correctly divided points. Perfect separability is achieved when $imp_\ell = 0$, whereas enforcing a positive importance threshold results in slightly lower but stable separability across all values of t_k. Values may extend beyond 100% due to variance, although separability itself is bounded.

Experimental Setup

We evaluate a grid of values for $\bar{\gamma}_t \in \{0.10, 0.15, \ldots, 1.00\}$ and $imp_\ell \in \{0.000, 0.001, 0.01, 0.1, 0.2, 0.3\}$, while keeping all other components of the architecture fixed. Explanation quality is measured through the mean number of nodes, reported together with its standard deviation. This setup allows us to isolate the individual and combined effects of $\bar{\gamma}_t$ and imp_l on explanation compactness.

Results and Discussion

The results reveal a clear and highly consistent pattern across both readability and local separability metrics. When $imp_\ell = 0$, explanations exhibit substantially higher readability values for all choices of t_k, accompanied by large variance (Figure 10.8). This indicates that, in the presence of a strict coverage constraint, a larger number of concepts are retained, leading to longer and less stable explanations. In this regime, local separability is maximised, with explanations achieving near-perfect division of the data (Figure 10.8).

Introducing a strictly positive importance threshold ($imp_\ell > 0$) leads to a sharp reduction in readability, with explanations becoming significantly more compact and markedly more stable. Notably, once imp_ℓ is active, readability remains constant across all tested values of t_k, indicating that the importance constraint dominates the explanation extraction process. A similar behaviour is observed for local separability: enforcing $imp_\ell > 0$ results in a modest decrease in division performance compared to the unconstrained case, but this value remains stable across the entire range of t_k.

Across both plots, variations in t_k have no observable effect once a positive

importance threshold is enforced, except for very small values where minor fluctuations can be observed. Overall, these results indicate that imp_ℓ acts as the primary control mechanism governing the trade-off between explanation simplicity and fidelity, effectively inducing a transition between exact but complex explanations and simpler, more compact ones. In contrast, t_k plays a secondary filtering role and does not require fine-tuning within the tested range.
This behaviour is consistent with the distinct roles of the two parameters. Since t_k operates at an early ranking stage, its effect is largely subsumed once the post-processing constraint imposed by imp_ℓ becomes active: regardless of the exact relevance threshold, additional concepts are iteratively incorporated until the required level of separability is achieved. As a consequence, variations in t_k do not materially alter the final explanation, provided that a sufficient pool of candidate concepts is available. Nevertheless, t_k remains a useful parameter, as it bounds the initial search space and prevents low-relevance concepts from being considered during post-processing. This contributes to computational efficiency and guards against the inclusion of spurious patterns, even though fine-grained tuning of t_k is not required in practice.

10.6.3 Summary of Findings

The analysis of the explanation mechanisms highlights the importance of structurally integrating explanation within the decision process. Variations in the extraction and aggregation procedures primarily affect global and local division metrics, revealing differences in explanation stability and class separation at the symbolic level. The results show that coherent explanatory patterns are preserved only when robustness information is consistently aligned with the discriminative objective. Simplified mechanisms reduce separation clarity, confirming that interpretability in STELLE is not an automatic by-product of classification, but the result of deliberate architectural design choices.

10.7 Overall Interpretation and Design Implications

The analyses presented in this chapter help clarify how the different components of STELLE contribute to its overall behaviour. Rather than identifying a single dominant factor, the results show that performance emerges from the interaction between the semantic structure of the concept space, the robustness-based embedding, and the explanation mechanism.

The experiments on concept set design indicate that the organisation of the STL basis has a direct impact on separation in the embedding space. Increasing the number of concepts or allowing greater structural diversity generally improves division metrics, but only up to a point. Beyond that regime, the gains become limited. This suggests that what matters is not the size of the concept set itself, but the extent to which it captures relevant and non-redundant semantic patterns.

The study of the kernel formulation confirms that quantitative robustness plays a central role. When robustness aggregation is simplified or altered, separation between classes becomes less stable. This shows that the embedding induced by robustness

is not interchangeable with a generic feature mapping, but is structurally tied to the discriminative properties of the model.

The architectural ablations further indicate that some components mainly affect the quality and stability of explanations rather than predictive behaviour alone. In other words, maintaining clear symbolic separation requires specific structural choices, even when classification accuracy appears relatively stable.

Finally, the analysis of the explanation mechanisms highlights that explanation quality is not an automatic consequence of prediction. Stable and coherent symbolic explanations depend on preserving alignment between robustness evaluation and class-specific scoring.

Overall, the results support the design choice of integrating symbolic representation, robustness semantics, and discriminative learning within a unified framework. The model is relatively robust to moderate variations in hyperparameters, but more sensitive to modifications that disrupt this integration. This suggests that the effectiveness of STELLE lies less in architectural complexity and more in the consistency between its representational and decision-making components.

Chapter 11

Conclusion

This thesis presented STELLE, a framework for time series classification that embeds Signal Temporal Logic (STL) directly into the learning process. By mapping raw trajectories and logic formulae into a shared semantic space, STELLE demonstrates that high classification accuracy does not require sacrificing formal interpretability. Unlike post-hoc methods, which attempt to approximate the reasoning of opaque "black box" models after training, STELLE treats interpretability as a fundamental architectural constraint. Its representations are constructed directly from logical concepts, ensuring that the generated explanations are not merely plausible rationalisations, but faithful descriptions of the model's internal decision-making.

A primary theoretical contribution is the trajectory–formula kernel, which extends the existing STL kernel framework to unify symbolic specifications and numerical data. This mathematical structure allows the model to measure the similarity between a continuous signal and a discrete logic rule via robustness semantics. Consequently, trajectories and formulae become directly comparable within a unified Hilbert space, providing a principled foundation for concept-based learning.

Architecturally, STELLE employs a transparent pipeline where a discriminability mechanism quantifies behavioural deviations and a learned relevance layer modulates feature importance for a linear classifier. This design facilitates the direct extraction of concise, readable local and global STL explanations without complex post-processing. Empirical benchmarks confirm that STELLE achieves performance competitive with state-of-the-art neural networks. Importantly, ablation studies and sensitivity analyses validate that this performance is robust, proving that deep learning-level accuracy can be achieved alongside the rigorous transparency of formal logic.

Crucially, the framework directly addresses the core limitations identified in the state-of-the-art analysis. Where post-hoc explainability methods approximate opaque decision boundaries, STELLE guarantees faithfulness by deriving predictions and explanations from the same semantic representation. Where deep time series models rely on latent embeddings without formal grounding, STELLE anchors representation learning in STL robustness semantics and, where prior symbolic approaches often produce rigid global rules with limited scalability, STELLE integrates temporal logic within a differentiable architecture capable of handling multivariate and multiclass settings. In doing so, the framework does not merely combine neural and symbolic components, but restructures their interaction around a shared formal foundation.

11.1 Contributions

This thesis introduces the STELLE framework, demonstrating that high predictive accuracy in time series classification doesn't need to come at the cost of formal interpretability. The impact of this work is defined by three primary contributions:

- *A functional-analytic foundation for temporal reasoning.* We establish a principled method for embedding raw time series data into the same Hilbert space as STL formulae. By extending the STL kernel framework to trajectories, we ensure that symbolic specifications and numerical data can be directly compared within a rigorous mathematical structure. This kernel acts as the computational engine of the framework, preserving behavioural alignment through robustness semantics.
- *The STELLE architecture.* We introduce a unified classification pipeline that integrates concept activation, discriminability analysis, and learned relevance. By treating interpretability as an architectural constraint rather than an afterthought, STELLE ensures faithfulness by construction: the explanations arise directly from the model's internal computations. This contrasts sharply with prevalent post-hoc approaches, which merely approximate the decision boundaries of opaque models.
- *An end-to-end framework for human-readable explanations.* Beyond the internal representation, we contributed a complete mechanism for extracting and simplifying logical rules. By applying robustness-induced threshold shifts, logic- and data-aware simplifications, these outputs are transformed from raw mathematical expressions into concise, readable temporal formulae suitable for domain experts.

11.2 Limitations

Whilst STELLE represents a significant step towards interpretable temporal learning, the current framework remains subject to some constraints:

- *Dependency on concept coverage.* The quality of STELLE's explanations depends fundamentally on the expressiveness of the STL concept set. Currently, the framework operates with a generic pool of formulae. If a critical temporal pattern is not represented in this initial set, the model cannot express it. While the current approach is domain-agnostic, future work could significantly enhance interpretability by explicitly integrating domain knowledge. Allowing experts to insert into the concept pool meaningful templates would narrow the semantic gap between raw data and explanation.
- *Evaluation of explanation quality.* Evaluating "interpretability" is inherently difficult. While we use proxy metrics like rule coverage and node count, these do not guarantee that a human finds the explanation helpful. True utility is subjective and context-dependent. To validate practical usefulness, a possible solution would be incorporate user studies with domain experts, moving beyond statistical proxies to qualitative feedback on decision support.

- *Multivariate interactions.* Complex interactions between multiple variables are naturally difficult to formalise and harder for humans to understand. While restricting the number of variables per formula limits expressiveness, it is a necessary constraint to prevent the logic from becoming unreadable.

11.3 Open Research Directions

The STELLE framework opens several promising avenues for future research:

- *Differentiable logic and parameter learning.* Currently, the framework relies on fixed, discretised thresholds for STL formulae. A powerful extension would be to make the temporal and value parameters of these formulae fully differentiable. By treating thresholds and time windows as learnable weights within the optimisation loop, the model could fine-tune decision boundaries. This would significantly reduce the combinatorial search space and allow the system to discover precise, data-driven cut-offs that the current discretisation might miss.

- *Interactive refinement with domain experts.* Because STELLE's "language" is symbolic, it offers a unique interface for human-in-the-loop learning. Unlike neural weights, which are abstract, an expert can read a learned formula and explicitly correct it (e.g., "this time window is too short"). Future work could develop active learning protocols where the model proposes a rule, the human critiques it, and the system updates its constraints accordingly.

- *Semantic transfer learning.* Logic is arguably more robust to domain shifts than raw statistical features. A temporal rule describing a mechanical fault (e.g., "pressure drops 5 seconds after vibration peaks") may hold true across different machines, even if the sensor noise profiles differ. Investigating whether STELLE's logic-based embeddings allow for zero-shot or few-shot generalisation across datasets is a high-potential future path.

- *Counterfactual generation.* The kernel used by STELLE naturally supports distance-based reasoning. This could be leveraged to generate counterfactuals automatically: given a time series classified as 'A', what is the minimal perturbation required to satisfy the formula for class 'B'? This would provide actionable debugging insights, moving beyond "why was this classified?" to "what needs to change?"

STELLE provides concrete evidence that the tension between predictive performance and formal interpretability is not necessarily inevitable. Rather than positioning logical reasoning and statistical learning as competing paradigms, this thesis shows that they can be integrated within a shared mathematical framework. By grounding representation learning in temporal logic semantics, STELLE reduces the reliance on post-hoc approximations, avoids the rigidity of purely symbolic rule-based models, and maintains competitive predictive accuracy without giving up semantic clarity.
Challenges remain, particularly with respect to scalability, concept coverage, and the evaluation of explanations from a human perspective. Nevertheless, the central message

of this work is clear: interpretability should be embedded into the design of learning systems, not appended after training. The framework presented here offers one concrete step in that direction, contributing to the development of more transparent and reliable approaches to time series analysis.

Declaration on Generative AI

During the preparation of this thesis, the author used Chat-GPT 5.1 for grammar and spelling check. No figures were generated by AI. The author reviewed and edited the content as needed and takes full responsibility for the thesis's content.

List of Figures

List of Tables

Bibliography

[1] Irene Ferfoglia, Simone Silvetti, Gaia Saveri, Laura Nenzi, and Luca Bortolussi. Guided by stars: Interpretable concept learning over time series via temporal logic semantics, 2025.

[2] Irene Ferfoglia, Gaia Saveri, Laura Nenzi, and Luca Bortolussi. Ecats: Explainable-by-design concept-based anomaly detection for time series. In Tarek R. Besold, Artur d'Avila Garcez, Ernesto Jimenez-Ruiz, Roberto Confalonieri, Pranava Madhyastha, and Benedikt Wagner, editors, *Neural-Symbolic Learning and Reasoning*, pages 175–191, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-71170-1.

[3] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

[4] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[5] Michael I. Jordan and Tom M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. doi: 10.1126/science.aaa8415.

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[7] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. URL https://arxiv.org/abs/1702.08608.

[8] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[10] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. doi: 10.1162/neco.1992.4.1.1.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[13] Drew McDermott, M. Mitchell Waldrop, B. Chandrasekaran, John McDermott, and Roger Schank. The dark ages of ai: A panel discussion at aaai-84. *AI Magazine*, 6 (3):122, Sep. 1985. doi: 10.1609/aimag.v6i3.494. URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/494.

[14] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[15] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[16] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

[17] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[18] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A%3A1010933404324.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 06 2009. doi: 10.1109/CVPR.2009.5206848. URL https://ieeexplore.ieee.org/abstract/document/5206848/.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, Red Hook, NY, USA, 2017. Curran Associates Inc.

[23] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on*

International Conference on Machine Learning, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[25] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014. URL http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL http://arxiv.org/abs/1502.03167. cite arxiv:1502.03167.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:213–252, 1990.

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

[30] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL http://arxiv.org/abs/1406.1078. cite arxiv:1406.1078Comment: EMNLP 2014.

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

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781. cite arxiv:1301.3781.

[34] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[35] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. URL http://arxiv.org/abs/2002.05709. cite arxiv:2002.05709Comment: ICML'2020. Code and pretrained models at https://github.com/google-research/simclr.

[36] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61 (10):36–43, September 2018. ISSN 0001-0782. doi: 10.1145/3233231. URL https://doi.org/10.1145/3233231.

[37] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1: 206–215, 05 2019. doi: 10.1038/s42256-019-0048-x.

[38] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *IEEE ICMLA*, pages 80–89, 2018.

[39] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *Proceedings of the 21st ACM SIGKDD*, pages 1721–1730, 2015.

[40] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *IEEE ICCV*, pages 843–852, 2017.

[41] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. 2019. Available at https://fairmlbook.org.

[42] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

[43] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *NIPS*, 2015.

[44] Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing*, 599:128096, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2024.128096. URL https://www.sciencedirect.com/science/article/pii/S0925231224008671.

[45] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *ACL*, 2019.

[46] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lilian Ma, Peter Mattson, and David R. So. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[47] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[49] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *ICML*, 2019.

[50] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019. doi: 10.1007/s10618-019-00619-1.

[51] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. URL http://arxiv.org/abs/1803.01271.

[52] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR*, 2018.

[53] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Explainable deep learning for time series: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[54] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, 4th edition, 2017.

[55] Granville Tunnicliffe Wilson. Time series analysis: Forecasting and control,5th edition, by george e. p. box, gwilym m. jenkins, gregory c. reinsel and greta m. ljung, 2015. published by john wiley and sons inc., hoboken, new jersey, pp. 712. isbn: 978-1-118-67502-1. *Journal of Time Series Analysis*, 37:n/a–n/a, 03 2016. doi: 10.1111/jtsa.12194.

[56] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. *European Business Intelligence Summer School*, pages 62–77, 2012.

[57] Yaguang Luo, Xiangrui Cai, Yixin Zhang, Jun Xu, and Xiaojie Yuan. Multivariate time series imputation with generative adversarial networks. *NeurIPS Workshop on Modeling Time Series*, 2018.

[58] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, December 2012. ISSN 0360-0300. doi: 10.1145/2379776.2379788. URL http://doi.acm.org/10.1145/2379776.2379788.

[59] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. Wiley, 3rd edition, 2019.

[60] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. *Encyclopedia of Machine Learning*, pages 257–258, 2011.

[61] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013. URL http://dblp.uni-trier.de/db/conf/esann/esann2013.html#AnguitaGOPR13.

[62] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer, Berlin, 2005. ISBN 3540262393 9783540262398. URL https://www.worldcat.org/title/new-introduction-to-multiple-time-series-analysis/oclc/1025413966?referer=br&ht=edition.

[63] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989. ISSN 0018-9219. doi: 10.1109/5.18626. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=18626&contentType=Journals+%26+Magazines&searchField%3DSearch_All%26queryText%3DA+Tutorial+on+Hidden+Markov+Models+and+Selected+Applications+in+Speech+Recognition.

[64] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35, 1960. doi: 10.1115/1.3662552. URL http://dx.doi.org/10.1115/1.3662552.

[65] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[66] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009. doi: 10.1145/1557019.1557122.

[67] Amaia Abanda, Usue Mori, and Jose A. Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019. doi: 10.1007/S10618-018-0596-4. URL https://scispace.com/papers/a-review-on-distance-based-time-series-classification-593jtszz1m.

[68] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 792–803. VLDB Endowment, 2004. ISBN 0120884690.

[69] Proximity forest 2.0: a new effective and scalable similarity-based classifier for time series - Data Mining and Knowledge Discovery — link.springer.com. https://link.springer.com/article/10.1007/s10618-024-01085-0#citeas. [Accessed 26-08-2025].

[70] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11, 2003. doi: 10.1145/882082.882086.

[71] Patrick Schäfer and Marcus Högqvist. Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527, 2012. doi: 10.1145/2168651.2168715.

[72] Aaron Boström and Anthony Bagnall. Shapelet transform classifiers for time series. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 292–301, 2017. doi: 10.1145/3097983.3098020.

[73] Matthew Middlehurst, James Large, and Anthony Bagnall. The hive-cote 2.0 time series classification framework. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 160–175. Springer, 2021. doi: 10.1007/978-3-030-86514-6_11.

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

[75] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[76] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017. doi: 10.1007/s10618-016-0483-9.

[77] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, September 2020. ISSN 1384-5810. doi: 10.1007/s10618-020-00710-y.

[78] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anastasios Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021. doi: 10.1145/3447548.3467401.

[79] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[80] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[81] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6845–6852, 4 2020. doi: 10.1609/aaai.v34i04.6165. URL https://ojs.aaai.org/index.php/AAAI/article/view/6165.

[82] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. Multirocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5):1623–1646, 2022.

[83] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[84] Yasamin Ghahremani and Vangelis Metsis. Time series embedding methods for classification tasks: A review. *arXiv preprint arXiv:2501.13392*, 2025. URL https://scispace.com/papers/time-series-embedding-methods-for-classification-tasks-a-1z49ph0e9gww.

[85] Christoph Molnar. *Interpretable Machine Learning*. 3 edition, 2025. ISBN 978-3-911578-03-5. URL https://christophm.github.io/interpretable-ml-book.

[86] J. Haugeland. *Artificial Intelligence: The Very Idea*. MIT Press, Cambridge, MA, 1985.

[87] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. ISSN 0885-6125. doi: 10.1023/A:1022643204877. URL https://doi.org/10.1023/A:1022643204877.

[88] Nativi S and De Nigris S. Ai standardisation landscape: state of play and link to the ec proposal for an ai regulatory framework. (KJ-NA-30772-EN-N (online)), 2021. ISSN 1831-9424 (online). doi: 10.2760/376602(online).

[89] High-Level Expert Group on AI. Ethics guidelines for trustworthy ai. Report, European Commission, Brussels, April 2019. URL https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[90] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 06 2017. ISSN 2044-3994. doi: 10.1093/idpl/ipx005. URL https://doi.org/10.1093/idpl/ipx005.

[91] Aug 2024. URL https://commission.europa.eu/news-and-media/news/ai-act-enters-force-2024-08-01_en.

[92] Implementation Timeline | EU Artificial Intelligence Act — artificialintelligenceact.eu. https://artificialintelligenceact.eu/implementation-timeline/. [Accessed 29-08-2025].

[93] Long awaited EU AI Act becomes law after publication in the EU's Official Journal | White & Case LLP — whitecase.com. www.whitecase.com/insight-alert/long-awaited-eu-ai-act-becomes-law-after-publication-eus-official-journal [Accessed 29-08-2025].

[94] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, 2017.

[95] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 279–288, 2019.

[96] Abeba Birhane. Algorithmic injustice: A relational ethics approach. *Patterns*, 2(2): 100205, 2021.

[97] Johanna Vielhaben, Sebastian Lapuschkin, Grégoire Montavon, and Wojciech Samek. Explainable ai for time series via virtual inspection layers. *Pattern Recognition*, 2024. doi: 10.1016/j.patcog.2024.110309. URL https://scispace.com/papers/explainable-ai-for-time-series-via-virtual-inspection-layers-4ps20uao7g.

[98] Vijay Arya, Diptikalyan Saha, Sandeep Hans, Amaresh Rajasekharan, and Tony Tang. Global explanations for multivariate time series models. 2023. doi: 10.1145/3570991.3570998. URL https://scispace.com/papers/global-explanations-for-multivariate-time-series-models-3kvyeuh8.

[99] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

[100] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017. URL https://arxiv.org/abs/1705.07874.

[101] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?". In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, August 2016. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145%2F2939672.2939778. Available at https://arxiv.org/pdf/1602.04938.pdf.

[102] David Alvarez-Melis and Tommi S. Jaakkola. On the robustness of interpretability methods, 2018. URL https://arxiv.org/abs/1806.08049.

[103] Panagiotis Papapetrou and Zed Lee. Interpretable and explainable time series mining. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2024. doi: 10.1109/dsaa61799.2024.10722788. URL https://scispace.com/papers/interpretable-and-explainable-time-series-mining-5bdfretpjf7z.

[104] Ephrem Tibebe Mekonnen, Luca Longo, and Pierpaolo Dondio. A global model-agnostic rule-based xai method based on parameterized event primitives for time series classifiers. *Frontiers in Artificial Intelligence*, 2024. doi: 10.3389/frai.2024.1381921. URL https://scispace.com/papers/a-global-model-agnostic-rule-based-xai-method-based-on-57hs1b4mc1yp.

[105] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3319–3328, 2017. URL https://arxiv.org/abs/1703.01365.

[106] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:51737170.

[107] Diana Mincu, Eric Loreaux, Shaobo Hou, Sebastien Baur, Ivan Protsyuk, Martin Seneviratne, Anne Mottram, Nenad Tomasev, Alan Karthikesalingam, and Jessica Schrouff. Concept-based model explanations for electronic health records. In *Proceedings of the Conference on Health, Inference, and Learning*, CHIL '21, page 36–46, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383592. doi: 10.1145/3450439.3451858. URL https://doi.org/10.1145/3450439.3451858.

[108] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, 3253:152–166, 01 2004. doi: 10.1007/978-3-540-30206-3_12.

[109] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems (FORMATS 2010)*, volume 6246 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2010. doi: 10.1007/978-3-642-15297-9_9.

[110] Laura Nenzi, Simone Silvetti, Ezio Bartocci, and Luca Bortolussi. A robust genetic algorithm for learning temporal specifications from data, 2018.

[111] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, page 1–10, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450339551. doi: 10.1145/2883817.2883843. URL https://doi.org/10.1145/2883817.2883843.

[112] Derek Long. A review of temporal logics. *The Knowledge Engineering Review*, 4 (2):141–162, 1989. doi: 10.1017/S0269888900004896.

[113] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57. IEEE, 1977. doi: 10.1109/SFCS.1977.32.

[114] Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. Learning signal temporal logic through neural network for interpretable classification, 2023. URL https://arxiv.org/abs/2210.01910.

[115] Harish Venkataraman, Derya Aksaray, and Peter Seiler. Tractable reinforcement learning of signal temporal logic objectives, 2020. URL https://arxiv.org/abs/2001.09467.

[116] Charu C. Aggarwal. First-order logic. In *Neural Networks and Deep Learning*, pages 109–133. Springer, 2021. doi: 10.1007/978-3-030-72357-6_5.

[117] Tarek Besold, Artur Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís Lamb, Priscila Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. *Chapter 1. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation.* 12 2021. ISBN 9781643682440. doi: 10.3233/FAIA210348.

[118] Wolfgang Rautenberg. *A Concise Introduction to Mathematical Logic.* Springer Publishing Company, Incorporated, 3rd edition, 2009. ISBN 1441912207.

[119] Dr Almeida, Maria Frade, Jorge Pinto, and Simão Sousa. *First-Order Logic*, pages 81–128. 01 2011. ISBN 978-0-85729-017-5. doi: 10.1007/978-0-85729-018-2_4.

[120] Lech Polkowski. *Logics for Computer and Data Sciences, and Artificial Intelligence.* 01 2022. ISBN 978-3-030-91679-4. doi: 10.1007/978-3-030-91680-0.

[121] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[122] G. E. Hughes and M. J. Cresswell. A new introduction to modal logic. *Studia Logica*, 62(3):439–441, 1996. doi: 10.4324/9780203028100.

[123] Clarence Irving Lewis. *Symbolic Logic.* The Century co., New York„ 1932.

[124] Saul A. Kripke. Semantical analysis of modal logic i. normal modal propositional calculi. *Journal of Symbolic Logic*, 31(1):120–122, 1966. doi: 10.2307/2270649.

[125] Edmund M Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986. doi: 10.1145/5397.5399.

[126] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* MIT Press, Cambridge, MA, USA, 2003. ISBN 0262562006.

[127] E. Allen EMERSON. Chapter 16 - temporal and modal logic. In JAN VAN LEEUWEN, editor, *Formal Models and Semantics*, Handbook of Theoretical Computer Science, pages 995–1072. Elsevier, Amsterdam, 1990. ISBN 978-0-444-88074-1. doi: https://doi.org/10.1016/B978-0-444-88074-1.50021-4. URL https://www.sciencedirect.com/science/article/pii/B9780444880741500214.

[128] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking.* The MIT Press, 2008. ISBN 026202649X. URL http://www.amazon.com/Principles-Model-Checking-Christel-Baier/dp/026202649X%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D026202649X.

[129] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42): 4262–4291, 2009. doi: 10.1016/j.tcs.2009.06.021.

[130] Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Nickovic. Survey on mining signal temporal logic specifications. *Information and Computation*, 289:104957, 2022. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2022.104957. URL https://www.sciencedirect.com/science/article/pii/S0890540122001122.

[131] Masaki Waga, Étienne André, and Ichiro Hasuo. Parametric timed pattern matching. *ACM Trans. Softw. Eng. Methodol.*, 32(1), February 2023. ISSN 1049-331X. doi: 10.1145/3517194. URL https://doi.org/10.1145/3517194.

[132] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[133] John McCarthy. Programs with common sense. *RLE and MIT Computation Center*, 1960.

[134] Ron Sun. An integrated connectionist and rule-based system for intelligent agents. *IEEE Transactions on Neural Networks*, 5(6):970–985, 1994.

[135] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119–165, 1994. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(94)90105-8. URL https://www.sciencedirect.com/science/article/pii/0004370294901058.

[136] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic ai: the 3rd wave. *Artif. Intell. Rev.*, 56(11):12387–12406, March 2023. ISSN 0269-2821. doi: 10.1007/s10462-023-10448-w. URL https://doi.org/10.1007/s10462-023-10448-w.

[137] Zishen Wan, Che-Kai Liu, Hanchen Yang, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Tushar Krishna, Yingyan Celine Lin, and Arijit Raychowdhury. Towards cognitive ai systems: a survey and prospective on neuro-symbolic ai. *arXiv preprint arXiv:2401.01040*, 2024. URL https://scispace.com/papers/towards-cognitive-ai-systems-a-survey-and-prospective-on-y7ss88qdij.

[138] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

[139] Emile van Krieken, Thiviyan Thanapalasingam, Jakub M. Tomczak, Frank van Harmelen, and Annette ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference, 2023. URL https://arxiv.org/abs/2212.12393.

[140] Lateb Nassim, Florence Sèdes, and Farida Bouarab-Dahmani. Toward compositional generalization with neuro-symbolic ai: A comprehensive overview. *Proceedings of EDIS*, 2024. doi: 10.1109/edis63605.2024.10783296.

[141] Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. Neuro-symbolic = neural + logical + probabilistic. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[142] Michelangelo Diligenti, Marco Gori, and Concetto Sacca. Semantic-based regularization for learning and inference. In *Artificial Intelligence*, volume 244, pages 143–165, 2017.

[143] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3749–3759, 2018.

[144] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813, 2020.

[145] Dan Xu, Elisa Ricci, and Nicu Sebe. Semantic bottleneck for computer vision tasks. In *British Machine Vision Conference (BMVC)*, 2018.

[146] Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2319–2328, 2017.

[147] Brandon Curtis Colelough and William C. Regli. Neuro-symbolic ai in 2024: A systematic review. *arXiv preprint arXiv:2501.05435*, 2025. URL https://scispace.com/papers/neuro-symbolic-ai-in-2024-a-systematic-review-na7s1wlve214.

[148] Moses Alabi and Sharima Moarales. Neuro-symbolic ai: Bridging the gap between symbolic and neural reasoning. 07 2024.

[149] Bikram Bhuyan, Amar Ramdane-Cherif, Thipendra P Singh, and Ravi Tomar. *Neuro-Symbolic AI: The Integration of Continuous Learning and Discrete Reasoning*, pages 29–44. 12 2024. ISBN 978-981-97-8170-6. doi: 10.1007/978-981-97-8171-3_3.

[150] Xin Zhang and Victor S. Sheng. Neuro-symbolic ai: Explainability, challenges, and future trends, 2024. URL https://arxiv.org/abs/2411.04383.

[151] Kushal Chauhan, Rishabh Tiwari, Jana von Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. *arXiv preprint arXiv:2212.07430*, 2022. URL https://scispace.com/papers/interactive-concept-bottleneck-models-190jx5dh.

[152] David
Debot, Pietro Barbiero, Francesco Giannini, Gabriele Ciravegna, Michelangelo

Diligenti, and Giuseppe Marra. Interpretable concept-based memory reasoning. *arXiv preprint arXiv:2407.15527*, 2024. URL https://scispace.com/papers/interpretable-concept-based-memory-reasoning-9vgmgbo0rbog.

[153] Satoshi Yamaguchi and K. Nishida. Explanation bottleneck models. *arXiv preprint arXiv:2409.17663*, 2024. URL https://scispace.com/papers/explanation-bottleneck-models-71ah8ztqk551.

[154] Jae Hee Lee, Sergio Lanza, and Stefan Wermter. From neural activations to concepts: A survey on explaining concepts in neural networks. *arXiv preprint arXiv:2310.11884*, 2023.

[155] Mert Yuksekgonul, Siyan Wang, and James Zou. Post-hoc concept bottleneck models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[156] Chih-Kuan Yeh, Been Kim, Alejandro Arrieta, Po-Sen Loh, Percy Liang, and Aleksander Madry. On the (in) completeness of representations learned by concept bottleneck models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[157] A. Bagnall, Jason Lines, Aaron George Bostrom, James Large, and Eamonn J. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660, 2016.

[158] Alberto Ruiz, Matt Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, December 2021. doi: 10.1007/s10618-020-00727-3.

[159] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33:917–963, 2018.

[160] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. *ArXiv*, abs/2403.14735, 2024.

[161] Hugues Turbé, Mina Bjelogrlic, Christian Lovis, and Gianmarco Mengaldo. Evaluation of post-hoc interpretability methods in time-series classification. *Nature Machine Intelligence*, 5:250–260, 2023.

[162] Andreas Theissler, Francesco Spinnato, Udo Schlegel, and Riccardo Guidotti. Explainable ai for time series classification: A review, taxonomy and research directions. *IEEE Access*, 10:100700–100724, 2022.

[163] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669, 2018. ISSN 2169-3536. doi: 10.1109/access.2017.2779939.

[164] Phongtharin Vinayavekhin, Subhajit Chaudhury, Asim Munawar, Don Joven Agravante, Giovanni De Magistris, Daiki Kimura, and Ryuki Tachibana. Focusing on what is relevant: Time-series learning and understanding using attention, 2018.

[165] Wendong Ge, Jin-Won Huh, Yu Rang Park, Jae-Ho Lee, Young-Hak Kim, and Alexander Turchin. An interpretable icu mortality prediction model based on logistic regression and recurrent neural networks with lstm units. In *AMIA Annual Symposium Proceedings*, volume 2018, page 460. American Medical Informatics Association, 2018.

[166] Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant Honavar. Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as informative time intervals, 2020.

[167] Wei Cai, Xiaomin Zhu, Kaiyuan Bai, Aihui Ye, and Runtong Zhang. An explainable dual-mode convolutional neural network for multivariate time series classification. *Knowledge-Based Systems*, 299:112015, 2024. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2024.112015.

[168] Francisco J. Baldán and José M. Benítez. Multivariate times series classification through an interpretable representation. *Information Sciences*, 569:596–614, 2021. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2021.05.024.

[169] Mario Refoyo and David Luengo. Multi-space: Multi-objective subsequence-based sparse counterfactual explanations for multivariate time series classification, 2024.

[170] Jacqueline Michelle Metsch, Philip Hempel, Miriam Cindy Maurer, Nicolai Spicher, and Anne-Christin Hauschild. Ensemble post-hoc explainable ai in multivariate time series: Identifying medical features driving disease prediction. *bioRxiv*, 2025. doi: 10.1101/2025.02.14.638219.

[171] Thies de Graaff, Michael Wild, Tino Werner, Eike Möhlmann, Stefan Seibt, and Benjamin Ebrecht. Increasing explainability in time series classification by functional decomposition. In Luca Longo, Sebastian Lapuschkin, and Christin Seifert, editors, *Explainable Artificial Intelligence*, pages 125–144, Cham, 2024. Springer Nature Switzerland.

[172] Giacomo Bergami, Emma Packer, Kirsty Scott, and Silvia Del Din. Towards explainable sequential learning, 2025.

[173] Jacqueline Höllig, Cedric Kulbach, and Steffen Thoma. Tsinterpret: A unified framework for time series interpretability, 2022.

[174] Udo Schlegel and Daniel A. Keim. A deep dive into perturbations as evaluation technique for time series xai. 2023.

[175] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable artificial intelligence (xai) on timeseries data: A survey, 2021.

[176] Ilija Šimić, Vedran Sabol, and Eduardo Veas. Xai methods for neural time series classification: A brief review, 2021.

[177] Pierre-Daniel Arsenault, Shengrui Wang, and Jean-Marc Patenaude. A survey of explainable artificial intelligence (xai) in financial time series forecasting. *ACM Computing Surveys*, 57(10):1–37, May 2025. ISSN 1557-7341. doi: 10.1145/3729531.

[178] Sara Mohammadinejad, Jyotirmoy V. Deshmukh, Aniruddh Gopinath Puranic, Marcell Vazquez-Chanlatte, and Alexandre Donzé. Interpretable classification of time-series data using efficient enumerative techniques. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control*, pages 9:1–9:10. ACM, 2020.

[179] Mohammed Kadous. Learning comprehensible descriptions of multivariate time series. 02 1970.

[180] Ruixuan Yan, Tengfei Ma, Achille Fokoue, Maria Chang, and A. Agung Julius. Neuro-symbolic models for interpretable time series classification using temporal logic description. *2022 IEEE International Conference on Data Mining (ICDM)*, pages 618–627, 2022.

[181] Giovanni Pagliarini, Simone Scaboro, Giuseppe Serra, Guido Sciavicco, and Eduard Ionel Stan. Neural-symbolic temporal decision trees for multivariate time series classification. In *Time*, 2024.

[182] Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. Tlinet: Differentiable neural network temporal logic inference, 2024.

[183] Ruixuan Yan, A. Agung Julius, Maria Chang, Achille Fokoue, Tengfei Ma, and Rosario A. Uceda-Sosa. Stone: Signal temporal logic neural network for time series classification. *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 778–787, 2021.

[184] Agnieszka Maria Jastrzebska, Gonzalo Nápoles, Yamisleydi Salgueiro, and Koen Vanhoof. Evaluating time series similarity using concept-based models. *Knowl. Based Syst.*, 238:107811, 2021.

[185] Christoph Obermair, Alexander Fuchs, Franz Pernkopf, Lukas Felsberger, Andrea Apollonio, and Daniel Wollmann. Example or prototype? learning concept-based explanations in time-series. In *Asian Conference on Machine Learning*, 2022.

[186] Gaurav R. Ghosal and Reza Abbasi-Asl. Multi-modal prototype learning for interpretable multivariable time series classification, 2021.

[187] Yupeng Wang, Jianghui Cai, Haifeng Yang, Chenhui Shi, Min Zhang, Jie Wang, Ran Zhang, and Xujun Zhao. Interpretable deep classification of time series based on class discriminative prototype learning. *Intelligent Data Analysis*, o(o): 1088467X251319188, 2025. doi: 10.1177/1088467X251319188.

[188] Mohammad Asadi, Vinitra Swamy, Jibril Frej, Julien Tuan Tu Vignoud, Mirko Marras, and Tanja Käser. Ripple: Concept-based interpretation for raw time series models in education. In *AAAI Conference on Artificial Intelligence*, 2022.

[189] Yuko Mizuhara, Akira Hayashi, and Nobuo Suematsu. Embedding of time series data by using dynamic time warping distances. *Systems and Computers in Japan*, 37(3):1–9, 2006. doi: https://doi.org/10.1002/scj.20486.

[190] Aaron Bostrom and Anthony Bagnall. *Binary Shapelet Transform for Multiclass Time Series Classification*. 07 2017. ISBN 978-3-662-55607-8. doi: 10.1007/978-3-662-55608-5_2.

[191] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series, 2022.

[192] Rachneet Kaur, Zhen Zeng, Tucker Balch, and Manuela Veloso. Lets-c: Leveraging language embedding for time series classification, 2024.

[193] Javier García-Sigüenza, Manuel Curado, Faraon Llorens-Largo, and Jose F Vicent. Self explainable graph convolutional recurrent network for spatio-temporal forecasting. *Machine Learning*, 114(1):2, 2025.

[194] Riccardo Spolaor. Verbal explanations of spatio-temporal graph neural networks for traffic forecasting, 2023.

[195] Gaia Saveri, Laura Nenzi, Luca Bortolussi, and Jan Křetínský. stl2vec: Semantic and interpretable vector representation of temporal logic, 2024. URL https://arxiv.org/abs/2405.14389.

[196] Luca Bortolussi, Giuseppe Maria Gallo, Jan Kretínský, and Laura Nenzi. Learning model checking and the kernel trick for signal temporal logic on stochastic processes. In *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022,Proceedings, Part I*, 2022.

[197] J. Hills, J Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28 (4):851–881, 7 2014. ISSN 1384-5810. doi: 10.1007/s10618-013-0322-1. URL https://ueaeprints.uea.ac.uk/id/eprint/42440/.

[198] Liang Tan and {Kenny K.N.} Chow. Coupling environmental affordances with schematic meaning: A matrix for designing embodied interaction in public spaces. In *Proceedings of Chinese CHI 2018 - 6th International Symposium of Chinese CHI, ChineseCHI 2018*, ACM International Conference Proceeding Series, pages 104–107. Association for Computing Machinery, April 2018. doi: 10.1145/3202667.3202682. 6th International Symposium of Chinese CHI, ChineseCHI 2018 ; Conference date: 21-04-2018 Through 22-04-2018.

[199] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. *Exact Discovery of Time Series Motifs*, pages 473–484. doi: 10.1137/

1.9781611972795.41. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.41.

[200] Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control*, 62(3): 1210–1222, 2017. doi: 10.1109/TAC.2016.2585083.

[201] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. doi: 10.1109/JAS.2019.1911747.

[202] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018, 2018.

[203] Anthony Bagnall, Michael Flynn, James Large, Jason Lines, and Matthew Middlehurst. *On the Usage and Performance of the Hierarchical Vote Collective of Transformation-Based Ensembles Version 1.0 (HIVE-COTE v1.0).* Springer International Publishing, 2020. ISBN 9783030657420. doi: 10.1007/978-3-030-65742-0_1.

[204] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11–12):3211–3243, September 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-06057-9.

[205] Angus Dempster, François Petitjean, and Geoffrey I Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[206] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction, 2013.

[207] Matthew Middlehurst, James Large, and Anthony Bagnall. The canonical interval forest (cif) classifier for time series classification. In *2020 IEEE International Conference on Big Data (Big Data)*, page 188–195. IEEE, December 2020. doi: 10.1109/bigdata50022.2020.9378424.

[208] Matthew Middlehurst, James Large, Gavin Cawley, and Anthony Bagnall. *The Temporal Dictionary Ensemble (TDE) Classifier for Time Series Classification.* Springer International Publishing, 2021. ISBN 9783030676582. doi: 10.1007/978-3-030-67658-2_38.

[209] Thach Nguyen, Severin Gsponer, Iulia Ilie, Martin O'reilly, and Georgiana Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Min. Knowl. Discov.*, 33 (4):1183–1222, July 2019. ISSN 1384-5810. doi: 10.1007/s10618-019-00633-3. URL https://doi.org/10.1007/s10618-019-00633-3.

[210] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+muse, 2018. URL https://arxiv.org/abs/1711.11343.

[211] Matthew Middlehurst, William Vickers, and Anthony Bagnall. Scalable dictionary classifiers for time series classification. In *Intelligent Data Engineering and Automated Learning – IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I*, page 11–19, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-33606-6. doi: 10.1007/978-3-030-33607-3_2. URL https://doi.org/10.1007/978-3-030-33607-3_2.

[212] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30, 09 2016. doi: 10.1007/s10618-016-0473-y.

[213] Giuseppe Bombara and Calin Belta. Offline and online learning of signal temporal logic formulae using decision trees. *ACM Trans. Cyber-Phys. Syst.*, 5(3), March 2021. ISSN 2378-962X. doi: 10.1145/3433994. URL https://doi.org/10.1145/3433994.

[214] Erfan Aasi, Cristian Ioan Vasile, Mahroo Bahreinian, and Calin A. Belta. Classification of time-series data using boosted decision trees. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1263–1268, 2021. URL https://api.semanticscholar.org/CorpusID:238259398.

[215] Erfan Aasi, Mingyu Cai, Cristian Ioan Vasile, and Calin Belta. Time-incremental learning of temporal logic classifiers using decision trees. In Nikolai Matni, Manfred Morari, and George J. Pappas, editors, *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pages 547–559. PMLR, 15–16 Jun 2023. URL https://proceedings.mlr.press/v211/aasi23a.html.

[216] Federico Pigozzi, Laura Nenzi, and Eric Medvet. BUSTLE: a Versatile Tool for the Evolutionary Learning of STL Specifications from Data. *Evolutionary Computation*, pages 1–24, 02 2024. ISSN 1063-6560. doi: 10.1162/evco_a_00347.

[217] Edgar A. Aguilar, Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Ničković. Mining specification parameters for multi-class classification. In Panagiotis Katsaros and Laura Nenzi, editors, *Runtime Verification*, pages 86–105, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-44267-4.

[218] Ruixuan Yan, Tengfei Ma, Achille Fokoue, Maria Chang, and Agung Julius. Neuro-symbolic Models for Interpretable Time Series Classification using Temporal Logic Description . In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 618–627, Los Alamitos, CA, USA, December 2022. IEEE Computer Society. doi: 10.1109/ICDM54844.2022.00072. URL https://doi.ieeecomputersociety.org/10.1109/ICDM54844.2022.00072.

[219] Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Learning interpretable rules for scalable data representation and classification. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 46(2):1121–1133, February 2024. ISSN 1939-3539. doi: 10.1109/tpami.2023.3328881. URL http://dx.doi.org/10.1109/TPAMI.2023.3328881.

[220] Yang Wang, Jiaqi Zhu, Miaomiao Li, Jiang Liu, Yilin Li, Yi Yang, Jiafan Li, and Hongan Wang. Learning reliable and intuitive temporal logic rules for interpretable time series classification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 3067–3078, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3737022. URL https://doi.org/10.1145/3711896.3737022.

[221] Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 38(4):1958–2031, April 2024. ISSN 1573-756X. doi: 10.1007/s10618-024-01022-1.

[222] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *Runtime Verification - Second International Conference, RV 2011, Revised Selected Papers*, volume 7186 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2011.

Appendix A

Concept Generation

This appendix reports the full procedure used to generate, instantiate, and select STL concepts. The material complements the high-level description provided in Section 8.2.2 and is included for completeness and reproducibility.

Template generation. To construct a flexible and interpretable vocabulary of temporal logic concepts, a large collection of STL formula templates is generated through grammar-based enumeration, exploiting Parametric Signal Temporal Logic (PSTL) [222]. PSTL extends STL by parameterising both the time bounds of temporal operators and the thresholds of inequality predicates. Given a PSTL template and a parameter configuration, a fully specified STL formula is obtained by instantiating all free parameters.

Formula templates are constructed recursively by applying logical and temporal operators up to a maximum of $M = 5$ syntactic nodes. To preserve interpretability, the number of distinct signal dimensions appearing in each formula is constrained to $n_{\text{vars}} \in \{1, 2, 3\}$. Atomic propositions take the form of simple threshold inequalities over individual signal components.

Template generation proceeds incrementally, starting from atomic propositions and expanding previously generated formulae using unary operators (e.g. $\mathbf{F}$, $\mathbf{G}$, $\neg$) and binary operators (e.g. $\wedge$, $\vee$, $\mathbf{U}$), subject to the imposed syntactic constraints. The full grammar-based enumeration procedure is reported in Algorithm 1.

Each template φ is instantiated over a finite grid of parameter configurations

$$\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\},$$

yielding a pool of fully specified STL formulae $\varphi(p)$. Parameter grids are defined independently for temporal bounds and predicate thresholds.

Robustness evaluation. All instantiated formulae are evaluated over a representative set of trajectories

$$\mathcal{T} = \{\tau_1, \ldots, \tau_{|\mathcal{T}|}\}$$

using the quantitative STL semantics. This yields a robustness matrix

$$S \in \mathbb{R}^{|\mathcal{P}| \times |\mathcal{T}|}, \qquad S(i, j) = \rho(\varphi(p_i), \tau_j),$$

where each row of S represents the robustness signature of a formula across the trajectory set.

Redundancy-based concept selection. Since the initial pool of instantiated formulae may contain behaviourally redundant concepts, a selection procedure is applied to obtain a compact and diverse concept set. Candidate formulae are considered sequentially and compared to the currently selected concepts using cosine similarity between robustness signatures. A formula is retained if its similarity to all previously selected concepts is below a fixed threshold $t \in (0, 1]$. This procedure incrementally builds a set of formulae whose robustness profiles are sufficiently distinct.

When multiple candidate formulae exhibit high similarity to existing concepts, preference is given to syntactically simpler formulae, retaining the structurally minimal representative. The selection process terminates once the desired number of concepts is reached.

The resulting concept set provides a compact, semantically diverse, and interpretable basis for embedding, classification, and symbolic explanation.

Algorithm 1 Algorithm for generating STL formulae templates

minipage=scale=0.95

Require: M, n_{vars}
Ensure: all_phis
  all_phis $\leftarrow$ [ ]
  all_phis.append(`generateAtomicPropositions`())
  for $2 \leq m \leq M$ **do**
    ▷ retrieve templates with $m-1$ nodes
    prev_phis $\leftarrow$`getPhisGivenNodes`$(m-1)$
    ▷ F, G, $\neg$
    unary_ops $\leftarrow$`expandbyUnaryOperators`(prev_phis)
    all_phis.append(unary_ops)
    ▷ all pairs $(l, r) : l + r = m, l \leq r$
    l_list, r_list $\leftarrow$`getPairsGivenSum`(m)
    for $(l, r) \in$ [l_list, r_list] **do**
      l_phis$\leftarrow$`getPhisGivenNodes`(l)
      r_phis$\leftarrow$`getPhisGivenNodes`(r)
      ▷ $\wedge$, $\vee$, U
      binary_ops$\leftarrow$`expandbyBinaryOpeators`(l_phis, r_phis)
      all_phis.append(binary_ops)

Appendix B

Synthetic Data Generation

This appendix provides a complete specification of the synthetic trajectory generation process used in the ablation studies (Section 10.1.1). The generator is designed to produce controllable multivariate time series with tunable class separability, enabling systematic evaluation of model behaviour under varying task difficulty.

B.1 Generation Algorithm

The synthetic data generator constructs multivariate time series trajectories through a compositional process that combines periodic, harmonic, trend, and noise components. Algorithm 2 presents the complete generation procedure.

The difficulty scale employs a two-sided interpolation scheme anchored at difficulty 5. For difficulties $d \in \{1, 2, 3, 4\}$, continuous parameters are computed as:

$$\theta(d) = \theta_{\text{easy}} + \frac{d-1}{4}\left(\theta_5 - \theta_{\text{easy}}\right),$$

where θ_{easy} denotes the difficulty 1 endpoint value and θ_5 denotes the difficulty 5 value.

For difficulties $d \in \{6, 7, 8, 9, 10\}$, parameters are computed as:

$$\theta(d) = \theta_5 + \frac{d-5}{5}\left(\theta_{\text{hard}} - \theta_5\right),$$

where θ_{hard} denotes the difficulty 10 endpoint value.

Categorical parameters (harmonics, trend type) follow discrete transition rules specified in Table B.1. This interpolation scheme ensures smooth difficulty progression whilst maintaining difficulty 5 as a reference point for all ablation experiments.

B.1.1 Parameter Specifications

Table B.1 specifies the endpoint and reference values for all generative parameters. These values were determined through iterative calibration to ensure monotonic accuracy degradation across baseline classifiers whilst maintaining appropriate task complexity at each difficulty level.

Each generative parameter affects class separability through a specific mechanism:

Table B.1: Difficulty-dependent parameter specifications. Difficulties 1–4 interpolate between easy and difficulty 5; difficulties 6–10 interpolate between difficulty 5 and hard.

Parameter	Symbol	Diff. 1 (easy)	Diff. 5 (medium)	Diff. 10 (hard)
Class separation	θ_{sep}	3.0	1.5	0.4
Within-class variance	θ_{var}	0.05	0.3	0.6
Temporal noise std.	$\theta_{\text{temp\_noise}}$	0.05	0.2	0.45
Fourier noise std.	$\theta_{\text{fourier\_noise}}$	0.0	0.125	0.3
Periodicity strength	θ_{period}	1.0	0.7	0.3
Frequency variation	$\theta_{\text{freq\_var}}$	0.05	0.35	0.7
Drift strength	θ_{drift}	0.01	0.06	0.15
Outlier probability	θ_{outlier}	0.0	0.04	0.1
Phase coherence	$\theta_{\text{coherence}}$	1.0	0.65	0.2
Categorical parameters:				
Use harmonics	$\theta_{\text{harmonics}}$	True	True	False ($d > 7$)
Number of harmonics	$\theta_{\text{n\_harm}}$	3	2	1
Trend type*	$\theta_{\text{trend\_type}}$	linear	linear	exponential

* Trend type follows: linear for $d \in \{1, \ldots, 7\}$, quadratic for $d \in \{8, 9\}$, exponential for $d = 10$.

- *Class separation* (θ_{sep}). Directly scales the amplitude differences between classes, controlling geometric distance in feature space.
- *Within-class variance* (θ_{var}). Controls stochastic variation within each class, increasing intra-class spread and overlap.
- *Temporal noise* ($\theta_{\text{temp\_noise}}$). Additive Gaussian noise in the time domain, degrading signal-to-noise ratio.
- *Fourier noise* ($\theta_{\text{fourier\_noise}}$). Complex noise added in the frequency domain, disrupting spectral structure.
- *Periodicity strength* (θ_{period}). Weight assigned to periodic components. Lower values reduce the dominance of class-characteristic frequencies.
- *Frequency variation* ($\theta_{\text{freq\_var}}$). Random perturbations to base frequencies within each class, reducing temporal consistency.
- *Drift strength* (θ_{drift}). Magnitude of non-stationary trend components, introducing class-dependent directional shifts.
- *Outlier probability* (θ_{outlier}). Fraction of trajectories subjected to extreme transformations, simulating label noise or distributional contamination.
- *Phase coherence* ($\theta_{\text{coherence}}$). Probability that phases are synchronised across variables. Lower values destroy multivariate dependencies.
- *Harmonics* ($\theta_{\text{harmonics}}$, $\theta_{\text{n\_harm}}$). Number of higher-order frequency components. Richer harmonic structure aids discriminability.

Table B.2: Observed test accuracy ranges by difficulty level. Values represent mean over five seeds across four baseline classifiers (ResNet, LSTM, Random Forest, Logistic Regression).

Difficulty	1	2	3	4	5
Accuracy range (%)	99.8–100.0	97.8–99.6	94.4–98.2	94.2–96.4	91.8–93.4
Difficulty	6	7	8	9	10
Accuracy range (%)	89.2–93.6	89.4–94.4	88.2–92.6	88.2–90.8	87.4–90.2

- *Trend type* ($\theta_{\text{trend\_type}}$). Functional form of drift (linear, quadratic, exponential). More complex trends are harder to model.

This multi-faceted approach ensures that difficulty reflects diverse aspects of time series classification complexity rather than a single confounding factor.

B.2 Observed Accuracy Ranges

Table B.2 reports the test accuracies achieved by baseline classifiers across all difficulty levels, averaged over five independent seeds. These results confirm successful calibration: accuracy decreases monotonically for all methods, with clear separation between difficulty endpoints. The accuracy ranges demonstrate that the generator produces tasks spanning from near-perfect classification (difficulty 1) to moderately challenging scenarios (difficulty 10), whilst maintaining learnable patterns at all difficulty levels. Difficulty 5 represents a balanced intermediate setting with baseline accuracies in the $92 - 93\%$ range, making it suitable for ablation studies where architectural differences can manifest without ceiling or floor effects.

Figure 10.2 (main text) demonstrates that the difficulty scale generalises across diverse architectural paradigms: deep learning models (ResNet, LSTM), ensemble methods (Random Forest), and linear classifiers all exhibit monotonic performance degradation. This cross-model consistency validates that difficulty reflects intrinsic dataset properties rather than model-specific biases.

Reproducibility Details

All experiments use the following fixed configuration:

- **Training set:** $N = 500$ trajectories, random seed $s = 0$
- **Test set:** $N = 100$ trajectories, random seed $s = 1$
- **Trajectory dimensions:** $n = 5$ variables, $T = 100$ time points
- **Number of classes:** $K = 3$ (balanced)
- **Difficulty level:** $d = 5$ (medium difficulty)

Both PyTorch and NumPy random number generators are initialised with the specified seeds to ensure full reproducibility. The implementation is available at https://github.com/ireneferfo/STELLE.

Algorithm 2 Synthetic Trajectory Generation

Require: Number of trajectories N, number of variables n, time points T, number of classes K, difficulty level $d \in \{1, \dots, 10\}$, random seed s
Ensure: Trajectory matrix $\mathbf{X} \in \mathbb{R}^{N \times n \times T}$, label vector $\mathbf{y} \in \{0, \dots, K-1\}^N$

Initialise: Set random seeds, $\mathbf{X} \leftarrow \mathbf{0}$, $t \leftarrow \text{linspace}(0, 1, T)$
Get parameters: $\theta \leftarrow \textsc{GetDifficultyParams}(d)$ ▷ Table B.1
Sample labels: $\mathbf{y} \leftarrow \text{Uniform}(\{0, \dots, K-1\}, N)$
Define class parameters:
for $k \in \{0, \dots, K-1\}$ **do**
    $f_k \leftarrow 2.0 + 1.5k$ ▷ Base frequency
    $a_k \leftarrow 1.0 + 0.3\theta_{\text{sep}} \cdot k$ ▷ Amplitude
    $\phi_k \leftarrow 2\pi k / K$ ▷ Phase offset
    $\psi_k \leftarrow k / \max(1, K-1)$ ▷ Shape parameter $\in [0, 1]$
for $i \in \{1, \dots, N\}$ **do**
    $k \leftarrow y_i$ ▷ Class of trajectory i
    isOutlier $\leftarrow \text{Bernoulli}(\theta_{\text{outlier}})$
    Sample class-conditional parameters:
    $f \leftarrow f_k \cdot (1 + \theta_{\text{var}} \cdot \mathcal{N}(0,1)) \cdot (1 + \theta_{\text{freq\_var}} \cdot \mathcal{N}(0,1))$
    $a \leftarrow a_k \cdot (1 + \theta_{\text{var}} \cdot \mathcal{N}(0,1))$
    for $v \in \{1, \dots, n\}$ **do**
        $f_v \leftarrow f \cdot (1 + 0.1v / \max(1, n))$ ▷ Variable-specific frequency
        Determine phase:
        if $\text{Uniform}(0,1) < \theta_{\text{coherence}}$ **then**
            $\phi \leftarrow \phi_k + 0.2v$ ▷ Coherent phase
        else
            $\phi \leftarrow \text{Uniform}(0, 2\pi)$ ▷ Random phase
        Build base signal: $s \leftarrow \mathbf{0} \in \mathbb{R}^T$
        $s \leftarrow s + \theta_{\text{period}} \cdot a \cdot \sin(2\pi f_v t + \phi)$ ▷ Primary periodic
        if $\theta_{\text{harmonics}}$ **then**
            for $h \in \{2, \dots, \theta_{\text{n\_harm}} + 1\}$ **do**
                $s \leftarrow s + 0.5\theta_{\text{period}} \cdot (a/h) \cdot \sin(2\pi h f_v t + h\phi)$
        Add non-periodic component:
        $w \leftarrow 1 - \theta_{\text{period}}$ ▷ Non-periodic weight
        if $\psi_k < 0.33$ **then**
            $g \leftarrow \exp(-(t - 0.5)^2 / 0.1)$ ▷ Gaussian shape
        else if $\psi_k < 0.67$ **then**
            $g \leftarrow |\sin(5\pi t)|$ ▷ Rectified sinusoid
        else
            $g \leftarrow 4t(1-t)$ ▷ Polynomial shape
        $s \leftarrow s + w \cdot a \cdot g$
        Add trend: $s \leftarrow s + \textsc{ComputeTrend}(t, v, k, \theta)$ ▷ Algorithm 3
        Add temporal noise: $s \leftarrow s + \theta_{\text{temp\_noise}} \cdot \mathcal{N}(\mathbf{0}, \mathbf{I}_T)$
        if $\theta_{\text{fourier\_noise}} > 0$ **then**
            Add frequency-domain noise:
            $\hat{s} \leftarrow \text{RFFT}(s)$ ▷ Real FFT
            $\hat{s} \leftarrow \hat{s} + \theta_{\text{fourier\_noise}} \cdot (\mathcal{N}(\mathbf{0}, \mathbf{I}) + i\mathcal{N}(\mathbf{0}, \mathbf{I}))$
            $s \leftarrow \text{IRFFT}(\hat{s}, T)$ ▷ Inverse real FFT
        if isOutlier **then**
            $s \leftarrow s \cdot (1 + 2\mathcal{N}(0,1)) + 3\mathcal{N}(0,1)$ ▷ Outlier transformation
        $\mathbf{X}_{i,v,:} \leftarrow s$
return $\mathbf{X}, \mathbf{y}$

Algorithm 3 Trend Computation

Require: Time vector $t \in \mathbb{R}^T$, variable index v, class k, parameters θ
Ensure: Trend signal $\tau \in \mathbb{R}^T$

$c \leftarrow \theta_{\text{drift}} \cdot v \cdot (k+1)$ ▷ Scaling coefficient
if $\theta_{\text{trend\_type}} =$ 'linear' **then**
    $\tau \leftarrow c \cdot t$
else if $\theta_{\text{trend\_type}} =$ 'quadratic' **then**
    $\tau \leftarrow c \cdot t^2$
else if $\theta_{\text{trend\_type}} =$ 'exponential' **then**
    $\tau \leftarrow c \cdot (\exp(t) - 1)$
else ▷ 'none'
    $\tau \leftarrow \mathbf{0}$
return τ

Appendix C

Formulae Manipulation and Post-processing

This appendix provides the full technical details of the post-processing operations applied to STL formulae during explanation generation. These procedures refine the symbolic expressions extracted from the model, ensuring they are semantically consistent, class–discriminative, and syntactically minimal. The appendix includes:

1. threshold–shift methods based on robustness landscapes,
2. logical and structural simplification rules,
3. data-aware reductions based on empirical predicate evaluations, and
4. proofs of semantic properties supporting the manipulations.

All definitions follow the STL syntax and robustness semantics introduced in Section 5.4.

C.1 Threshold Adjustment and Polarity Correction

Given an STL formula φ extracted as part of a local or global explanation for a trajectory $\mathbf{x}$ and target class $\hat{y}$, we refine its decision boundary by analysing robustness patterns across classes. The goal is to enhance interpretability and discriminativeness without modifying temporal structure.

Let

$$r_{\text{target}} = \rho(\varphi, \mathbf{x}), \qquad \mathcal{R}_k = \{\rho(\varphi, \tau) : \tau \in \mathcal{T}_k\},$$

where $\mathcal{T}_k$ is the set of trajectories belonging to class k. The robustness values of all opposing classes form

$$\mathcal{R}_{\text{opp}} = \bigcup_{k \neq \hat{y}} \mathcal{R}_k.$$

We then examine whether the target robustness lies above or below the opposing distribution. If the majority of values in $\mathcal{R}_{\text{opp}}$ lie above r_{target}, the formula's polarity is reversed:

$$\varphi \leftarrow \neg\varphi, \qquad r_{\text{target}} \leftarrow -r_{\text{target}}, \qquad \mathcal{R}_{\text{opp}} \leftarrow \{-r : r \in \mathcal{R}_{\text{opp}}\}.$$

Algorithm 4 Postprocess formulae based on robustness separation

Require: Set of formulae Φ, target class $\hat{y}$, input trajectory $\mathbf{x}$, trajectories by class $\{\mathcal{T}_1, \ldots, \mathcal{T}_{K-1}\}$
Ensure: Postprocessed formulae Φ'

```
Φ' ← ∅
for φ ∈ Φ do
    r_target ← ρ(φ, x)
    for k = 0 to K − 1 do
        if k ≠ ŷ then
            R_k ← [ρ(φ, τ) for τ ∈ T_c]
        else
            R_k ← [r_target]
    R_opp ← ⋃_{k≠ŷ} R_k
    n_< ← #{r ∈ R_opp | r < r_target}
    n_> ← #{r ∈ R_opp | r > r_target}
    if n_> > n_< then
        φ ← ¬φ
        r_target ← −r_target
        R_opp ← [−r | r ∈ R_opp]
    r_closest ← max{r ∈ R_opp | r < r_target}
    s ← (r_target + r_closest)/2
    φ' ← RescaleThresholds(φ, −s)
    if ρ(φ', x) < 0 then
        φ' ← ¬φ'
    append φ' to Φ'
return Φ'
```

To create a sharper discriminatory boundary, we compute the robustness value r_{closest} of the opposing class that is nearest but still below r_{target}, and shift the formula's predicate thresholds by:

$$s = \frac{1}{2}(r_{\text{target}} + r_{\text{closest}}).$$

Thresholds are uniformly adjusted by $-s$; if this adjustment results in negative robustness for $\mathbf{x}$, the formula is negated (preserving consistency). A complete specification is provided in Algorithm 4.

C.2 Linearity of Robustness under Threshold Shifts

We provide the full proof of the proposition invoked in the main text, which establishes that uniformly adjusting all thresholds in an STL formula by a constant δ induces a linear shift of $-\delta$ in the robustness values computed over any trajectory. This property enables us to normalise robustness distributions, such as centring a collection of values $\{\rho_1, \rho_2, \ldots, \rho_n\}$ computed over different trajectories, by modifying the formula

thresholds.

Preposition. By uniformly modifying the thresholds of the variables in an STL formula φ by a constant δ, the robustness values $\rho(\varphi, \tau, t)$ for any trajectory τ will shift linearly by $-\delta$.

Proof. We now provide proof of this statement for all STL. By proving that the Preposition holds true for every Atom and STL operator, the proof comes naturally for more complex structures.

Step 1: Atomic Predicates
Consider the atomic predicate $\pi(\mathbf{x}) = \big(f_\pi(\mathbf{x}) \geq 0\big)$ and a trajectory τ.
Its robustness value is:

$$\rho(\pi, \tau, t) = f_\pi(\tau(t))$$

If we adjust the threshold by a constant δ, this corresponds to modifying the predicate to

$$\pi(\mathbf{x}) = \big(f_\pi(\mathbf{x}) + \delta \geq 0\big) = \big(f_\pi(x) \geq -\delta\big)$$

The new robustness value becomes:

$$\rho'(\pi, \tau, t) = f_\pi(\tau(t)) - \delta = \rho(\pi, \tau, t) - \delta$$

Thus, a uniform adjustment by δ decreases the robustness linearly by $-\delta$.

Step 2: Temporal operators
Next, let's consider the impact on more complex formulae built from atomic predicates.

Let's start from *Negation*, whose definition is:

$$\rho(\neg\varphi, \tau, t) = -\rho(\varphi, \tau, t)$$

The a uniform adjustment by δ results in:

$$\begin{aligned}\rho'(\neg\varphi, \tau, t) &= -\left(\rho(\varphi, \tau, t) - \delta\right) \\ &= -\rho(\varphi, \tau, t) + \delta \\ &= \rho(\neg\varphi, \tau, t) + \delta\end{aligned}$$

This shows that the robustness shifts linearly by $-\delta$.

For *Conjunction*, defined as

$$\rho(\varphi_1 \wedge \varphi_2, \tau, t) = \min\left(\rho(\varphi_1, \tau, t), \rho(\varphi_2, \tau, t)\right)$$

after uniform adjustment becomes:

$$\rho'(\varphi_1 \wedge \varphi_2, \tau, t) = \min\left(\rho(\varphi_1, \tau, t) - \delta, \rho(\varphi_2, \tau, t) - \delta\right)$$

Since $\min(a - \delta, b - \delta) = \min(a, b) - \delta$, we have:

$$\rho'(\varphi_1 \wedge \varphi_2, \tau, t) = \rho(\varphi_1 \wedge \varphi_2, \tau, t) - \delta$$

Again, the robustness shifts linearly by $-\delta$.

To prove the property for the *Until* operator, we again start from its definition

$$\rho(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2, \tau, t) = \max_{t' \in [t+a, t+b]} \left(\min \left(\rho(\varphi_2, \tau, t'), \min_{t'' \in [t, t']} \rho(\varphi_1, \tau, t'') \right) \right)$$

After uniform adjustment:

$$\rho'(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2, \tau, t) = \max_{t' \in [t+a, t+b]} \left(\min \left(\rho(\varphi_2, \tau, t') - \delta, \min_{t'' \in [t, t']} \rho(\varphi_1, \tau, t'') - \delta \right) \right)$$

Simplifying using properties of $\max$ and $\min$:

$$\rho'(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2, \tau, t) = \rho(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2, \tau, t) - \delta$$

Hence, the robustness again shifts linearly by $-\delta$.

Finally, the proof for *Eventually* and *Globally* comes naturally from their definitions and the properties of $\max$ and $\min$.

This concludes the proof that for any formula φ, the uniform adjustment of thresholds by a constant δ consistently shifts linearly the robustness value $\rho(\varphi, \tau, t)$. □

C.3 Logical and Data-Aware Simplifications

A summary of the simplifications applied can be found in Table C.1.

Logical simplifications. This simplification procedure applies structural equivalences to STL formulae. Double negation is eliminated via $\neg(\neg\varphi) \to \varphi$, while De Morgan laws transform $\neg(\varphi \wedge \psi)$ into $\neg\varphi \vee \neg\psi$ and $\neg(\varphi \vee \psi)$ into $\neg\varphi \wedge \neg\psi$. Redundant binary expressions are collapsed using $\varphi \wedge \varphi \to \varphi$ and $\varphi \vee \varphi \to \varphi$. The simplifier also reduces combinations like $\varphi \wedge (\varphi \vee \psi)$ and $\varphi \vee (\varphi \wedge \psi)$ to φ.

Temporal nesting is flattened. Considering I and J as two generic temporal intervals, $\mathbf{G}_I(\mathbf{G}_J(\varphi))$ becomes $\mathbf{G}_{I+J}(\varphi)$, and similarly $\mathbf{F}_I(\mathbf{F}_J(\varphi)) \to \mathbf{F}_{I+J}(\varphi)$. When both children of an until operator are equal, the formula $\varphi \ \mathbf{U}_I \ \varphi$ simplifies to φ. This procedure does not depend on data and only rewrites formulae based on their syntactic structure.

Data-aware simplifications. This method uses the evaluation of atomic predicates over a trajectory set $\mathcal{T}$ to simplify STL formulae contextually. Each atom $\pi(\mathbf{x})$ is evaluated across all considered trajectories and at all time steps. If the predicate holds (resp. fails) for all samples at a given t, it is marked as true (resp. false). Otherwise, it is marked as undefined at that time step.

Given this truth map, each subformula is recursively reduced. An atom that is uniformly true (resp. false) is replaced by $\top$ (resp. $\bot$). Boolean simplification is then applied, collapsing expressions like $\top \wedge \varphi$ to φ and $\bot \vee \varphi$ to φ.

Table C.1: STL formulae simplifications

Logical simplifications	
Original	*Simplified*
$\neg(\neg\varphi)$	φ
$\neg(x \leq c)$	$x > c$
$\neg(\mathbf{F}_I(\varphi))$	$\mathbf{G}_I(\neg\varphi)$
$\neg(\mathbf{G}_I(\varphi))$	$\mathbf{F}_I(\neg\varphi)$
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1 \wedge \neg\varphi_2$
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1 \vee \neg\varphi_2$
$\varphi \wedge \varphi$	φ
$\varphi \vee \varphi$	φ
$\varphi \wedge (\varphi \vee \psi)$	φ
$\varphi \vee (\varphi \wedge \psi)$	φ
$\varphi \wedge (\varphi \wedge \psi)$	$\varphi \wedge \psi$
$\varphi \vee (\varphi \vee \psi)$	$\varphi \vee \psi$
$\mathbf{G}_I(\mathbf{G}_J(\varphi))$	$\mathbf{G}_{I+J}(\varphi)$
$\mathbf{F}_I(\mathbf{F}_J(\varphi))$	$\mathbf{F}_{I+J}(\varphi)$
$\varphi\ \mathbf{U}_I\ \varphi$	φ

Data-aware simplifications	
Original	*Simplified*
$\neg\texttt{True}$	$\texttt{False}$
$\neg\texttt{False}$	$\texttt{True}$
$\texttt{True} \wedge \varphi$	φ
$\texttt{False} \wedge \varphi$	$\texttt{False}$
$\texttt{True} \vee \varphi$	$\texttt{True}$
$\texttt{False} \vee \varphi$	φ
$\mathbf{G}_{[0,\infty)}(\texttt{True})$	$\texttt{True}$
$\mathbf{G}_{[0,\infty)}(\texttt{False})$	$\texttt{False}$
$\mathbf{F}_{[0,\infty)}(\texttt{True})$	$\texttt{True}$
$\mathbf{F}_{[0,\infty)}(\texttt{False})$	$\texttt{False}$
$\varphi\ \mathbf{U}_{[0,b]}\ \texttt{True}$	φ
$\varphi\ \mathbf{U}_{[a,b]}\ \texttt{True}$, $a \neq 0$	$\mathbf{G}_{[0,a]}\varphi$
$\varphi\ \mathbf{U}_I\ \texttt{False}$	$\texttt{False}$
$\texttt{True}\ \mathbf{U}_I\ \varphi$	$\mathbf{F}_I(\varphi)$
$\texttt{False}\ \mathbf{U}_I\ \varphi$	$\texttt{False}$

Appendix D

Additional Results

This appendix presents a comprehensive collection of local explanations generated by the proposed framework across different domains. The objective is to demonstrate the versatility of the method in extracting interpretable Signal Temporal Logic (STL) formulae that characterise instance-level behaviours.

We provide supplementary visualisations for two distinct datasets: the multivariate *Maritime* surveillance dataset and the univariate *ECG200* medical dataset.

D.1 Maritime Local Explanations

This section visualises local explanations generated for the Maritime dataset. As detailed in the main text, this dataset consists of multivariate time series data where variables x_0 and x_1 correspond to the planar coordinates (latitude and longitude) of vessels entering the harbour.

Figures D.1 and D.2 display representative trajectories alongside their computed STL explanations. The plots illustrate how the method captures specific geometric corridors and temporal windows that define the safe and typical behaviour of maritime traffic, distinguishing it from anomalous patterns.

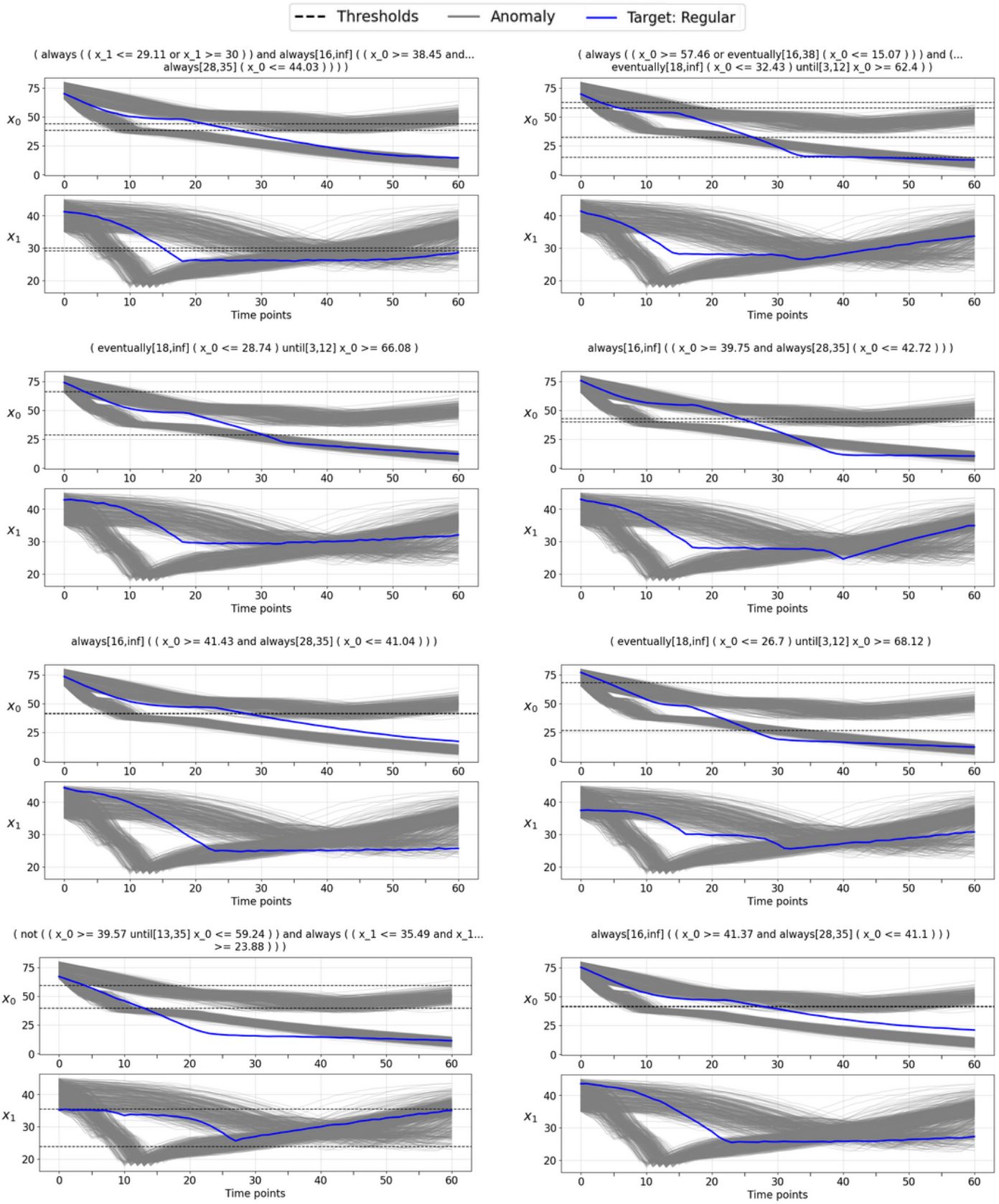


Figure D.1: Local explanations for regular (Class 0) maritime trajectories. The target trajectory is highlighted in blue, while the opposing anomalous class is shown in grey. The extracted STL formulae identify the standard entry corridors and spatial bounds that characterise safe harbour access.

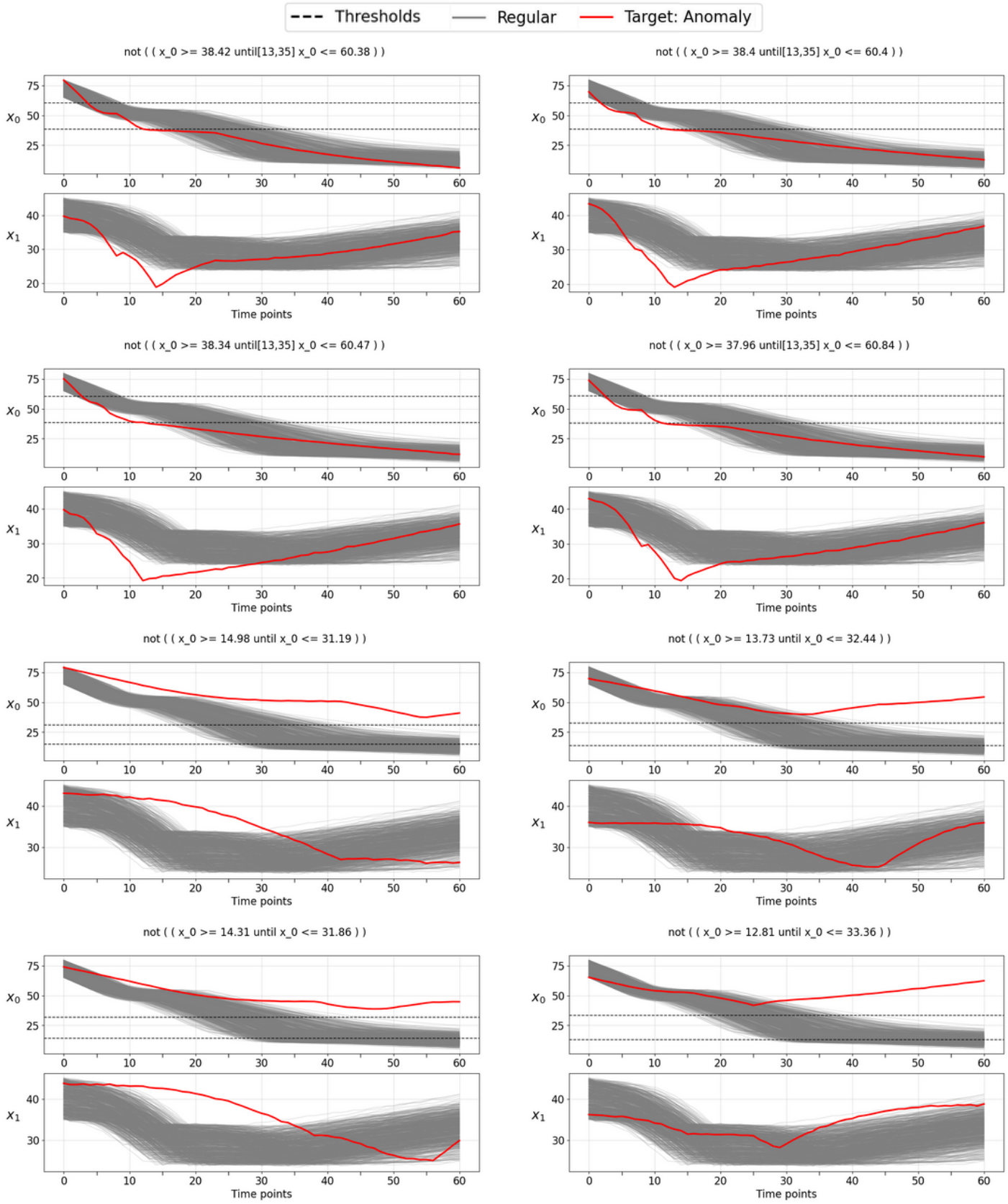


Figure D.2: Local explanations for anomalous (Class 1) maritime trajectories. The target trajectory is highlighted in red. The formulae capture specific deviations from the standard path, such as spatial violations or abnormal temporal ordering of waypoints.

D.2 ECG200 Local Explanations

We further examine the explanations generated for the *ECG200* dataset, which traces the electrical activity of the heart during individual beats. The task involves distinguishing between Normal heartbeats (Class 0) and those indicative of Myocardial Infarction (Class 1).

Figures D.3 and D.4 present the time-resolved signals and their associated logical explanations. In this context, the STL formulae serve to identify morphological features, such as peak amplitude stability or the timing of the recovery phase, that are clinically relevant.

For normal heartbeats (Figure D.3), the explanations typically enforce specific amplitude ranges and transition timings consistent with a healthy cardiac cycle. Conversely, for pathological cases (Figure D.4), the formulae highlight the absence of expected peaks or the presence of abnormal plateaus that characterise the infarction.

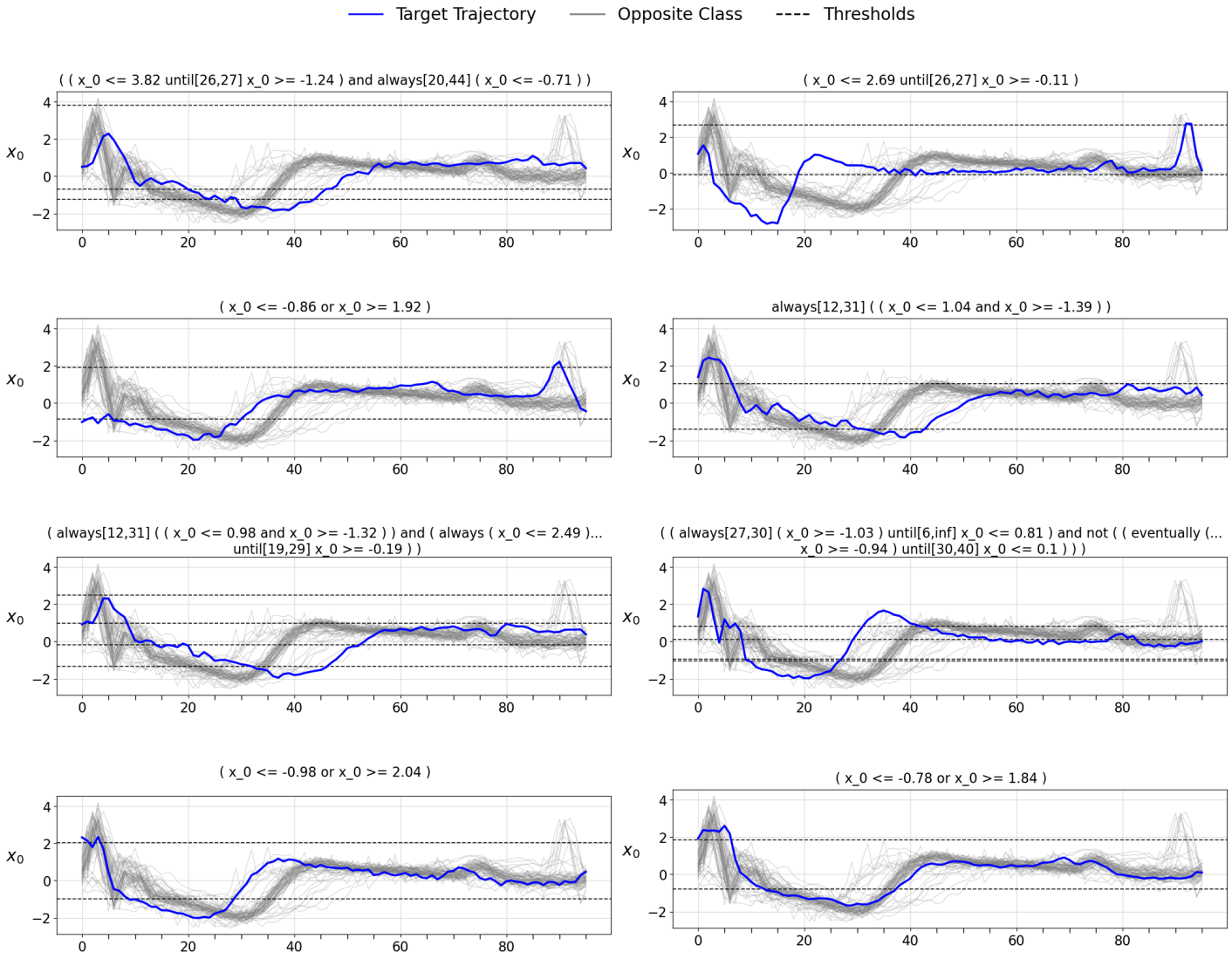


Figure D.3: Local explanations for Normal (Class 0) heartbeats. The blue line represents the target healthy beat, distinguished from the Myocardial Infarction class (grey background) by formulae that capture the characteristic high-amplitude peaks and proper temporal evolution of the wave.

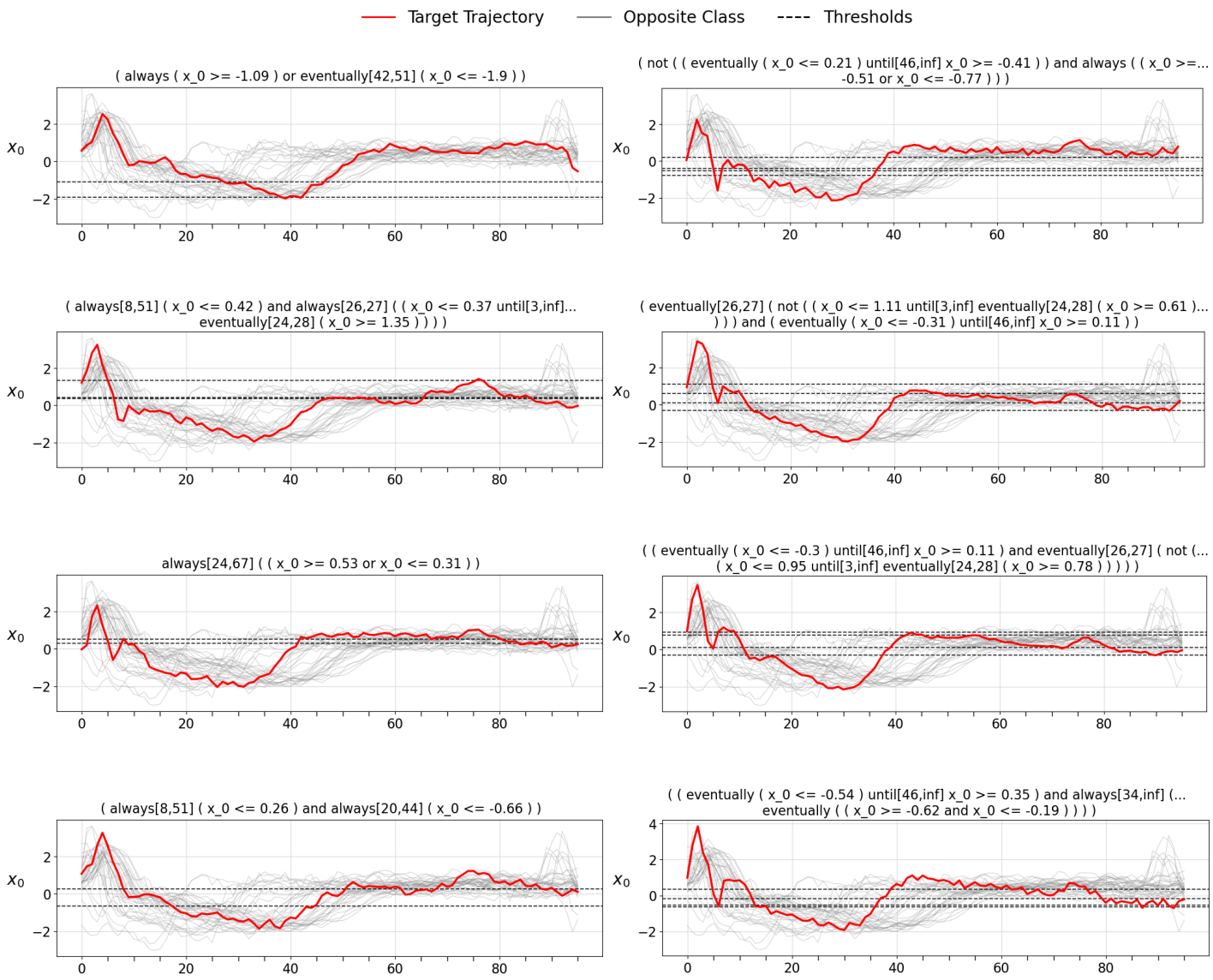


Figure D.4: Local explanations for Myocardial Infarction (Class 1) heartbeats. The red line traces the pathological beat. The generated explanations identify these anomalies through constraints that describe the suppressed amplitude or distorted morphology typical of the condition.

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

La borsa di dottorato è cofinanziata dal progetto “iNEST – Interconnected Nord-Est Innovation Ecosystem”, ECS_00000043, parte del programma di ricerca dell’ecosistema dell’innovazione a valere sulle risorse del Piano Nazionale per la Ripresa e Resilienza (PNRR), M4C2 - Investimento 1.5 Creazione e rafforzamento di “Ecosistemi dell’innovazione per la sostenibilità”, finanziato dall’Unione Europea, NextGenerationEU - CUP J43C22000320006