Universiteit Leiden

Master Computer Science

Generating Natural Language Explanations for Deep Time Series Classification Models

Name:	Hedieh Pourghasem
Student ID:	3903818
Date:	2025-08-14
Specialisation:	Data Science
1st supervisor:	Prof. dr. Suzan Verberne
2nd supervisor:	Dr. Niki van Stein

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgment

This thesis is a significant personal achievement for me. Taking on a topic with a considerable research gap presented unique challenges, and I'm truly thankful for the support and guidance I received through the way.

My sincere thanks go to my first supervisor, Dr. Suzan Verberne, for her invaluable insights and dedicated supervision. Our regular meetings were greatly contributory and helpful for the progress of this work. Her guidance not only helped me navigate the challenges of my research, but also significantly improved my critical thinking and research skills. I'm especially grateful for her understanding and compassion during personal difficulties; her support was genuinely heartwarming during some dark days.

I also want to thank my second supervisor for her helpful feedback and assistance. And a special note of appreciation to her PhD student, Qi Huang, whose technical guidance and recommendations were incredibly beneficial throughout the process.

To my beloved partner, thank you for your unwavering love and daily support during my master program. This thesis report feels like a shared milestone in the journey we began two years ago when we moved to build our life together in the Netherlands.

Lastly, but certainly not least, I extend my deepest gratitude to my family and close friends, especially my mother and father. Their love and support were always deeply felt, even from a distance. I truly believe that my entire academic path, and my belief and passion in science, were profoundly influenced by them and I consider myself very fortunate to have had their guidance and encouragement in my life.

Abstract

In this thesis, we propose a post-hoc explanation pipeline for time series classification models that generates local natural language explanations (NLEs) based on attribution-values from attribution-based explanation methods. Our goal is to make the logic behind classifier model's predictions more interpretable and understandable for both expert and non-expert users by translating model reasoning into textual explanation. We focus on exploring how large language models (LLMs) can be guided to produce faithful and useful explanations when given attribution values of a sample input.

To do this, we design a pipeline that includes classification, attribution generation, categorizing attribution values, and explanation generation using both rule-based templates and prompt-based LLMs. We evaluate our approach on two datasets: a synthetic time series dataset with ground-truth attribution labels, and a real-world human activity recognition dataset. Our experiments show that LLMs can produce logically correct, faithful and helpful explanations, but only when given specific hints about which time steps have the most influential attribution values, and when the prompts are carefully engineered to guide the model's reasoning.

On a synthetic dataset, we systematically evaluated different prompt structures and showed that passing filtered attribution inputs significantly improved explanation faithfulness and completeness. On the real-world dataset, we conducted another round of prompt engineering to address LLM hallucinations caused by general knowledge about real-world sensor data. This led to a final prompt version that included all necessary rules and instructions to guide the LLM. We then performed a user study with expert and non-expert participants to assess the quality and usefulness of both explanation types. The results show that LLM-based explanations were consistently preferred over rule-based ones and also achieved higher ratings in terms of clarity and helpfulness. Non-expert users particularly benefited from the natural language structure, and majority found the combination of textual and visual explanations the most helpful explanation modality for understanding the model's behavior.

These findings suggest that attribution-guided LLMs can be an effective method for explaining time series model decisions, but they require careful input structuring and prompt design to avoid hallucinations and preserve faithfulness and completeness. This work highlights both the potential and the limitations of using natural language explanations for interpreting time series classifier decisions.

Contents

1 Introduction

Time series classification (TSC) is an important task in many fields, such as the medical domain, technical systems, and finance [1]. In these fields, distinguishing between different patterns or categories and identifying anomalies is an important problem and has a significant impact. While ML-based time series classification models' performance is improving, the models become more complex and the reasoning behind these models' predictions is becoming harder to understand for both engineers and end users. This means that they require an explanation regarding their decision-making process to become understandable and interpretable by humans.

The lack of explainability or interpretability of the model is a challenge. It can restrict the use of ML-based time series classification models in real-world applications. Improving explainability helps users understand the decision process, so they can better judge the quality and value of the outcome, especially when making high-stakes decisions [2].

The goal of eXplainable AI (XAI) research is to make AI systems more comprehensible and transparent to humans without sacrificing performance [3]. Since around 2019, the XAI field for time series classification has gained attention. XAI studies introduce a variety of models with different explanation types, such as attribution-based, attention-based, shaplets, prototypes, counterfactuals, etc. [4]. These methods usually highlight points, subsequences, or instances of a time series that lead to a specific prediction, helping to understand the reasoning behind the performance of a classification model.

A newly emerging type of explanation method is natural language explanations (NLEs). NLEs describe the model's decision-making process using natural language text that is easily understandable for humans. The development of NLE methods has improved a lot due to the emergence and improvements of large language models (LLMs) [5]. NLEs are especially useful when domain experts need to understand why the model gives a certain result for a specific sample (local explanation). For example, a doctor using a time series model to help diagnose a patient may need to understand the reasoning of the model. Many of these domain experts do not have technical knowledge, so they cannot easily understand the mathematical or visual output of common explanation methods.

Figure 1 illustrates this challenge. The upper subplot shows multivariate time-series input, while the lower subplot presents the corresponding attribution values. These values, generated by attribution-based explanation methods, are often visualized using heatmaps, scatter plots, or other graphical representations. However, they are often difficult to interpret without having a technical understanding of XAI. They lack contextual reasoning, clarity, and guidance for non-technical users.

NLEs have been investigated in the context of explainability for computer vision models and vision-to-text tasks [6, 7]. DeViL [7] generates text explanations for images using a transformer model as "translator" to convert learned weight vectors into prefix tokens for an image captioning task. These works leverage image captioning datasets where there are input samples (images) connected to detailed textual labels, This provides a way to connect internal weights of a vision model to natural language concepts. In contrast, time series classification typically lacks such datasets with descriptive text, making direct translation of model representations into explanations infeasible. This requires a different approach which we aim to suggest in our study.

Recent work in NLP, such as EvalxNLP [8], has explored benchmarking explainability methods using LLMs to verbalize attribution values. For time series data, TSXplain [9] represents

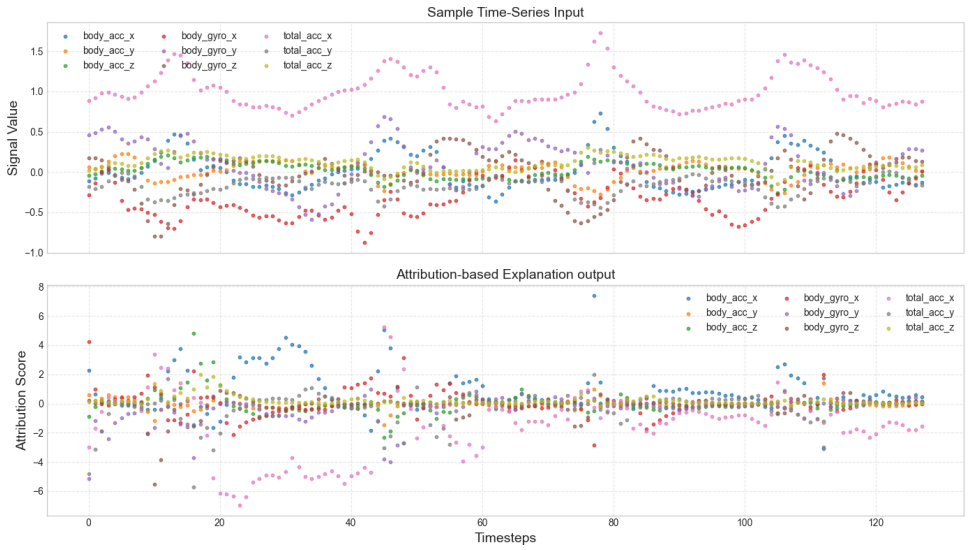

Figure 1: Top: Example of multivariate time-series sensor input. Bottom: Attribution-based explanation output showing per-feature importance scores across time.

an early effort to generate NLEs for anomaly detection through rule-based (template-based) approaches. However, these works either do not focus on natural language generation for the TSC problem or rely on rigid templates. Specifically, no research has yet investigated the use of LLMs to generate natural language explanations for the predictions of time series classification models.

We propose to explain deep learning-based time series classifiers using natural language through the well-known attention-based explanation methods. We apply attention-based methods to a trained deep learning classifier to get attention values for each point in the time series. These attribution values are then processed to generate natural language explanations via two strategies: a rule-based template approach and an LLM-based approach. We carefully design the prompts to ensure the LLM generates faithful, clear, and informative explanations. Moreover, we customize explanation styles according to the target user's expertise (expert and non-expert), and the quality of these explanations is assessed through human judgment. We conduct this evaluation both on the explanations alone and in combination with attribution visualizations, by asking participants which version they found more helpful.

In this work, we address three main research questions:

1. How to build a pipeline that can explain logical reasoning behind deep time series classification models in form of natural language?

2. Can LLMs be guided to produce accurate and helpful NLEs based on attribution values from deep time series classifiers?

3. What are the relative strengths and weaknesses of rule-based versus LLM-generated explanations for time series classification?

4. How do expert and non-expert users perceive and evaluate different forms of explanations, and what explanation styles and modalities are most helpful for each group?

In summary, this work addresses a significant gap in the current literature by exploring the field of natural language explanations for time series classification models. We propose and inves-

tigate multiple strategies for generating natural language explanations for time series models. We develop a modular pipeline that combines deep learning classifiers, attribution-based explanation methods, and large language models to generate an interpretable and human-readable explanation. Finally, we evaluate the resulting explanations in terms of their helpfulness, and alignment with the model reasoning.

2 Related work

Time series classification (TSC) has been an active field of research for decades. This is due to the presence of univariate and multivariate time series data, a time-ordered sequence of values, in many real-world applications and increase of temporal data availability.
As a result, a wide range of TSC methods from classic distance-based techniques to deep neural networks have been developed. In this section, we first discuss important developments in TSC methods. Then we categorize and review different XAI methods used to explain TSC models, including methods. We also review previous work focused on generating natural language explanations (NLE). Finally, we discuss important criteria for evaluating NLE based on previous studies. This sets the stage for our work on explaining TSC models through natural language explanations.

2.1 Time Series Classification Methods

A useful way to understand the evolution of TSC methods is through a taxonomy based on the type of data representation they use as proposed in a study by Middlehurst et al. (2024) [10], TSC algorithms can be broadly categorized into: distance-based, feature-based, interval-based, shapelet-based, dictionary-based, convolution-based, and deep learning-based approaches.
Early TSC approaches relied on distance-based methods with a time series specific distance measure. One of the these well known methods is the one-nearest neighbor classifier with the Dynamic Time Warping (DTW) [11] metric which is often considered as a baseline for time series classification. DTW is a similarity measure which aligns two time series by warping the time axis, helps in matching similar patterns. A limitation of this approach is the high time and space complexity so it doesn't scale well to complex multivariate time series [12].
Other early approaches introduced more structured techniques, such as shapelet-based classifiers that use discriminative subsequences, and dictionary-based models like WEASEL [13] that convert time series into bags of symbolic words using symbolic approximation. These traditional models often relied on hand-crafted features or fixed similarity measures, which limited their adaptability to complex, noisy, or highly variable time series. These limitations lead to a shift toward deep learning based models that learn features directly from time series data without manual feature engineering.

2.1.1 CNN and RNN-Based Methods

More recently, deep learning approaches demonstrated strong performance on both univariate and multivariate TSC tasks. Many deep learning models are better suited to handle time series data, such as Recurrent neural networks (RNNs), convolutional neural networks (CNNs) which are designed in a way to capture the temporal dependencies.
Early deep learning methods were inspired by the success of CNNs and residual networks (ResNets) [14] in other fields such as image processing. InceptionTime [15] is a widely-used deep learning model for time series classification that applies multiple convolution filters of different sizes in parallel to capture features at different temporal resolutions. It uses an ensemble of residual networks to improve accuracy and generalization. Later, H-InceptionTime [16] (introducing by adding handcrafted filters) and LITETime [17] (a lightweight variant) have been proposed as improved versions of InceptionTime. Although effective at detecting local temporal/spatial patterns, CNN models are limited in their ability to capture long-range dependencies and the full sequence order.

RNNs are another kind of neural models designed to handle sequential data such as time series, due to their internal memory structure that allows them to capture temporal dependencies. RNNs can be used in a many-to-one architecture for TSC problem. Using vanilla RNNs for TSC was first proposed in [18], however, vanilla RNNs suffer from vanishing and exploding gradient problems, making them difficult to train and perform well in long time series. This led to the development of long short-term memory (LSTM) networks [19] and gated recurrent units (GRUs) [20], which introduced gating mechanisms to better handle long-term dependencies. The disadvantage of RNN-based models is that they are computationally expensive and their capability to capture long-range dependencies is limited. To address these challenges, attention-based and transformer architectures have become popular.

2.1.2 Transformer-based methods

Transformer architectures, originally introduced for natural language processing [21]. Transformers use a self-attention layer, which allows models to selectively focus on relevant parts of the input sequence. This allows transformers to capture long-range dependencies more effectively than RNNs and process entire sequences in parallel, improving both scalability and performance.
According to a survey on time series transformers [22], the transformer is shown to be effective in various time series classification tasks due to its ability to learn long-term dependencies. For example, the Time Series Transformer (TST) adapts the vanilla transformer by adding learned positional embeddings and processing multivariate inputs. This model is able to handle both temporal order and cross-channel dependencies in time series [23]. Similarly, Gated Transformer Networks (GTN) use a two-tower attention structure. In these models one tower focuses on temporal patterns and the other captures relationships between different variables. This approach achieve promising results on multivariate TSC tasks [24].

2.2 XAI in Time Series Classification

XAI methods have several distinct charachteristics: ante-hoc vs. post-hoc, global vs. local, and model-agnostic vs. model-specific. XAI algorithms can be ante-hoc meaning the classification model to be inherently interpretable such as decision trees while post-hoc methods are separated from the classification model and being applied on the classification model after training. Furthermore, global explanations focus on capturing and describing the overall decision-making logic of the model, whereas local explanations aim to explain model decisions for each input sample. Model-agnostic methods are independent of the structure and type of classifiers, while model-specific techniques try to explain the internal architecture of particular models.
Theissler et al. [4] categorize XAI approaches for time series classification into three main categories:

- Time-point-based methods assign a relevance score or weight to each time point in a time series, indicating its contribution to the model's decision. Two known approaches for obtaining these relevance scores are attribution-based and attentions-based methods.

  Attribution-based techniques, such as SHAP (Shapley Additive Explanations) [25] and Integrated Gradients [26] , analyze the input-output relationship by perturbing input values or computing gradient-based relevance scores.

Attentions-based methods, are applicable in TSC models which use the attention mechanisms such as transformer models and attention-augmented RNNs. As discussed in subsection 2.1, attention models are suitable for sequential data and boost performance by learning to reliably learn from long range dependencies. The attention mechanisms highlight important time steps by learning weighting sequence during model training. Attention bease approaches often visualize as heatmaps and are hard to interpret in many cases [27].

- Subsequence-based methods identify critical subsequences within a time series that significantly influence classification decisions. These methods use either shapelets (short, discriminative subsequences that are not necessarily contiguous, often termed improper subsequences) or patches/patterns (contiguous segments, also known as proper subsequences) to extract segments from time series that strongly influence on classification of specific class labels.

  Shapelet-based methods directly integrate these distinctive segments into the classification process, commonly through simple and interpretable models like decision trees. In contrast, pattern-based methods look for meaningful, repeated sections within the time series, using methods Symbolic Aggregate approXimation (SAX)[28] recognizing frequently occurring patterns relevant to classification outcomes.

- Instance-based methods use entire series or instances to explain model behavior. Instance-based explanations aim to compare a given time series sample with representative instances (prototypes or counterfactuals) to justify a classification decision. Prototype-based methods explain predictions by finding and showing similar examples from the training data. In other words, they help to understand a model's decision by comparing it directly with familiar examples from past data. Counterfactual explanations, on the other hand, identify minimal changes in an input instance that would alter the prediction of the model, providing understanding of the model decision boundaries.

Each of these approaches offers distinct insights and is beneficial depending on the specific interpretability needs of the application. In this work we use multiple state-of-the-art attribution-based explainers for post-hoc local explainability. These include Temporal Integrated Gradients [29], Temporal Occlusion [30], DynaMAsk [31], and ExtremalMask [32] to generate point-based local explanation from deep neural networks. In the following, we provide a brief overview of how each of these methods operates.

2.3 Attribution-Based Methods Applied in This Study

- **Temporal Integrated Gradients (TIG)** [29]: TIG is an extension of the classic Integrated Gradients [33] method specifically designed for time series data. In time series, using future information to explain a decision at time t is illogical. TIG avoids this by cropping the sequence at each time step t to explain the model's decision. Then, it computes the integrated gradients by interpolating only the last time point x_t, while the previous time steps are unchanged. For each feature i at time t, it computes the attribution by summing the gradients between a baseline input x' and the actual input x.The attribution is given by Equation 1:

$$\mathsf{TIG}_{ti} = (x_{ti} - x'_{ti}) \int_0^1 \frac{\partial F(x'_{1:t} + \alpha(x_{1:t} - x'_{1:t}))}{\partial x_{ti}} \, d\alpha \tag{1}$$

where F is the model's prediction function, x_{ti} is the actual input for feature i at time t, and x'_{ti} is the corresponding baseline value. Finally, it aggregates the feature-wise attributions and normalizes the result to obtain a relevance score for the entire time step.

- **Temporal Occlusion** [30]: Temporal Occlusion modifies the standard Occlusion [34] adapted for time series. It is a perturbation-based attribution method. At each time step t, only the final observation x_t is perturbed, while all previous inputs $x_{1:t-1}$ remain unchanged. The model's output is then recomputed using the perturbed input, and the difference from the original prediction is used to estimate the importance of that time step. Temporal Occlusion is most useful when gradients are unavailable or unreliable, such as in models with vanishing gradients.

- **DynaMask** [31]: DynaMask is another perturbation-based method that generates local feature attributions for time series models by learning a mask over the input. It fits a continuous-valued mask matrix $M \in [0,1]^{T \times D}$, where each element indicates the importance of feature i at time t. The method perturbs the input sequence dynamically using the temporal neighborhood of each input. The mask is optimized to minimize the shift in the model's prediction under perturbation.

- **ExtremalMask** [32]: ExtremalMask improves DynaMask by learning both an attribution mask and a perturbation generator. It constructs a saliency mask $M \in [0,1]^{T \times D}$ over the input and uses a neural network to generate replacements for the masked entries. So, instead of relying on fixed replacements like noise or averages it is generating context-aware perturbations. The mask and perturbation network are trained together to minimize the difference between the model's output on the original and perturbed inputs.

It is important to note that explanations generated from these XAI methods are mostly in form of visualizations or statistical and numerical output, which provide important information for computer scientists and researchers but might not be intelligible and understandable for end users who do not have a computer science or math related background. This limitation underscores the importance of developing more accessible and user-friendly forms of explanation, motivating researchers to explore using natural language as a medium to describe how models work.

2.4 Natural language explanations

Natural Language Explanations (NLEs) represent a new and emerging area in the field of XAI. NLE methods translate the technical and often abstract outputs of machine learning models into human-readable textual descriptions, thus significantly improving interpretability, especially for non-expert users. Most research in this area has focused on applications in computer vision and natural language processing.
In the computer vision domain, several methods have been proposed to generate NLEs from visual features. Kayser et al. [35] introduced e-ViL, a benchmark dataset for evaluating NLEs in

vision-language tasks. Sammani et al. [6], proposed NLX-GPT to generate NLEs in multimodal vision and tasks, using LLM. A more recent approach for generating NLEs in computer vision is the DeViL (Decoding Vision features into Language) [7]. DeViL employs a transformer-based model to decode intermediate vision features, extracted from different layers of a convolutional neural network (CNN), into natural language descriptions. It trains a transformer network on image captioning datasets to map vision feature maps into tokens that can be interpreted by a pre-trained LLM. The novelty of this approach lies in using the text labels from image captioning tasks to train a transformer model that acts as a kind of "translator," mapping the learned weights in the CNN model to a hidden vector space that is understandable by a pre-trained LLM. This enables the generation of natural language explanations for a given input of the model's feature values.

In the domain of NLP, the structure of input data and the token level attribution information, have enabled the development NLE techniques. Feldhus et al. [5] introduced the task of Saliency Map Verbalization in NLP, where token-level attributions from feature attribution methods are translated into natural language explanations. They proposed and evaluated two approaches for generating these explanations: a template-based method and an instruction-based method using LLMs.

In another work in NLP, Dhaini et al. [8] proposed EvalxNLP, a benchmarking framework for evaluating post-hoc explainability methods in NLP models. EvalxNLP integrates different attribution methods and evaluates them using metrics such as faithfulness, plausibility, and complexity. In addition to these benchmarking features, the framework also has an LLM-based module that generates natural language explanations to help users better understand both the attribution scores and the evaluation metrics. Although EvalxNLP integrates an LLM-based module to verbalize attribution scores and evaluation results, it does not focus on the natural language explanation (NLE) task itself. There is no exploration of prompt design or systematic evaluation of the generated NLE. Although both of these works (by Feldhus and Dhaini) focus on text classification, their methodology and evaluation results offer valuable insight for extending NLE generation to other domains, such as time series.

In contrast to these developments, only a few studies explored NLEs in time series data. One notable early work that addresses this gap is TSXplain [9]. TSXplain focuses on generating NLEs for anomaly detection in time series data. TSXplain integrates deep neural networks with statistical feature extraction techniques to generate useful textual explanations for both experts and novice users. The method begins by identifying influential data points in a time series through an influence tracking mechanism (TSViz [36]), then characterizing these points using statistical measures. These characteristics are then combined into template-based natural language descriptions, resulting in text explanations of detected anomalies. While its explanations rely on fixed templates, the method shows a way on how structured data characteristics can be translated into text. This sets a nice foundation for future work on LLM-driven NLEs in time series domains.

2.5 Evaluation

Evaluating the quality and effectiveness of explanation methods is a critical but challenging task. Although many work has proposed evaluation metrics for time series explainers, different methods are often evaluated using different task-specific criteria. XTSC-Bench [37] introduces a standardized benchmarking framework specialized for evaluating explanation methods in TSC. It proposes four core metrics including faithfulness, robustness, complexity, and reliability

to quantitatively assess the performance of explanation methods. Faithfulness measures how accurately the explanation reflects the model's decision-making process. Robustness evaluates the stability of an explanation when the input is slightly perturbed. Complexity quantifies how many features the explanation uses. Simpler explanations with fewer features are preferred, as they are generally easier to understand. Reliability focuses on how well the explanation highlights truly informative regions of the input, based on known ground truth.
The metrics in XTSC-Bench are mainly designed for numerical outputs and assume a clear link between input features and explanations. However, NLEs provide abstract human-readable text that doesn't always match specific features or time points. So, evaluating NLEs for end users needs different methods that focus on how correct, clear, and useful the text explanations are. Among existing works, TSXplain [9] is the only work tailored to assessing NLEs for time series. TSXplain conducted a qualitative user study involving two distinct user groups to evaluate the quality of the generated textual explanations. Novice users, with limited knowledge of time-series data, focused on high-level understanding, while expert users sought more detailed and technical explanations. The participants rated the explanations based on subjective evaluation metrics including relevance, sufficiency, and correctness. Additionally, expert users also assessed the meaningfulness and their overall satisfaction. This evaluation setup helped capture diverse perspectives on the clarity and usefulness of the natural language explanations for end users.
In a study by Leiter et al. [38] on metrics for the evaluation of natural language explanations, they suggest that it is important to focus on things like faithfulness, plausibility, and transparency in the evaluation. This idea matches well with the challenges we face in evaluating NLEs for time series classification, where usual metrics designed for numerical explanation outputs don't really capture how good the NLEs are. These ideas can be helpful for designing better and more understandable ways to evaluate textual explanations in our research.

Despite these advancements, there remains a significant gap in the development of NLEs for time series tasks, specifically for time series classification. One of the key challenges is the lack of labeled datasets that include textual explanations for time series labels, which makes training and evaluating such models difficult. Moreover, to the best of our knowledge, there is no prior work that generate NLEs for time series data using the text generation ability and general knowledge of LLMs. This represents a promising and largely unexplored direction in XAI research. In this work, we aim to address this gap by investigating the use of LLMs to generate natural language explanations specifically for time series classification tasks.

3 Experimental Methodology

We propose a post-hoc explanation pipeline that generates local, NLEs of a time series classification model. Our goal is to transform the logic behind a model reasoning on a specific sample input, into human-understandable text explanation for both expert and non-expert users.
We define expert users as individuals with a background in computer science or related fields who are familiar with time series models and XAI methods. Non-expert users are those without technical expertise in XAI, such as domain professionals or general users, who need more basic explanations to understand model behavior.
The core of our approach is the use of attribution-based explanation methods. These methods are chosen for their ability to assign importance scores to each input point, showing its influence on the model's decision. These attribution scores are used to generate NLEs through two strategies: a template-based approach, which we use as a baseline, and a prompt-based LLM approach. The template method provides a structured and controlled baseline and the use of LLMs to generate can result in more clear, flexible, and contextful explanations.

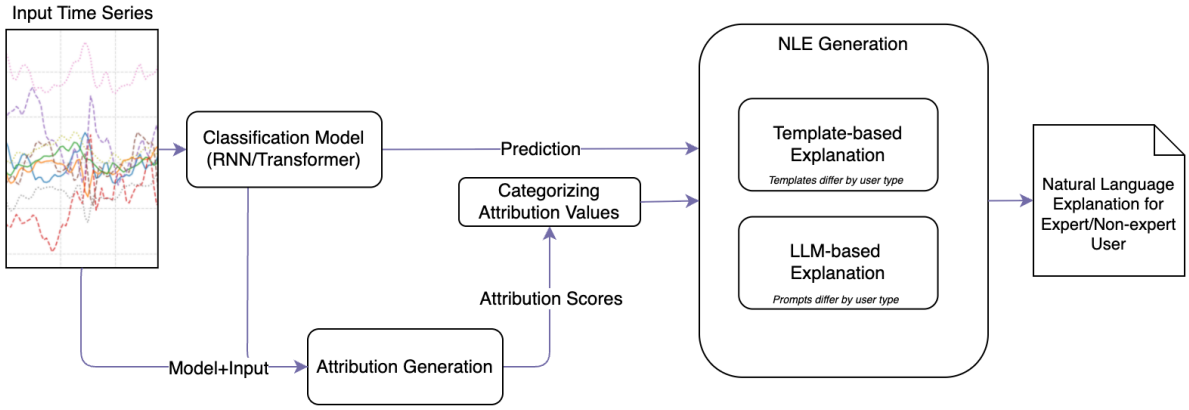

Figure 2: End-to-end pipeline for generating local natural-language explanations of multivariate time series classifications.

Figure 2 illustrates the overall pipeline. The designed pipeline consists of four stages:

1. **Classification Models:** We train two deep learning models, including a simple RNN and a transformer-based architecture on a multivariate time series classification dataset. These are the models whose predictions we aim to explain.

2. **Attribution Generation:** We apply attribution methods on each sample with the trained classification model to obtain attribution scores per feature and per time step. These scores serve as the basis for NLE generation.

3. **Categorizing Attribution Values:** We categorize the raw attribution values into three levels: strongly influential, moderately influential, and not influential. These levels help to structure the input in the explanation stage. In the template-based approach, this controls which time steps and features are mentioned. In the LLM-based method, they are optionally included in prompts to guide the model's reasoning.

4. **Natural Language Explanation Generation:**

- **Template-based Explanation:** A rule-based method that constructs explanations from categorized attribution scores. We design separate templates for expert and non-expert users, adapting the level of details and technicality.
- **LLM-based Explanation:** A method in which a structured summary of prediction, attribution scores, and other relevant information is passed to an LLM. We explore multiple prompt formats and evaluate how phrasing and relevant context affect the faithfulness, completeness, and usefulness of the explanation.

We conducted experiments on two multivariate time series classification datasets: one synthetic dataset with ground truth attribution labels, and one real-world human activity recognition dataset without such labels. The synthetic dataset was used as a pre-study in a controlled setting to benchmark attribution methods and explore different prompt structures. We evaluated the final NLE output on this dataset ourselves. The real-world dataset helps us test how well the pipeline works in a practical scenario, including user evaluation with both expert and non-expert users.
The following subsections describe the datasets, the implementation of each pipeline stage, and the evaluation methodology in each stage.

3.1 Datasets

Selecting a suitable dataset for this problem was a challenge. An ideal dataset would be a time series classification dataset that includes labels or brief explanations in natural language along with the class labels. This would make it possible to train and evaluate models capable of generating textual explanations for a TSC task. However, such datasets are currently not available. As a result, we focused on finding a time series classification dataset with either clear, human-interpretable patterns or ground truth attribution labels, which could provide a reliable foundation for assessing explanation faithfulness.
We found an artificial dataset introduced by Tonekaboni et al. [30], named the Spike dataset. This dataset was specifically designed to evaluate time series explainability methods. Provides ground-truth attribution labels, allowing for quantitative assessment of the explanation accuracy of attribution values. This dataset was originally proposed for a time series forecasting task, but with small changes we were able to use it for a time series classification.
To further validate our method, we also conducted experiments on a real-world dataset. We sought a labeled time series classification problem that was simple and interpretable, allowing human and non-expert evaluation of generated explanations. For this, we selected the UCI HAR dataset [39], which contains labeled human activity data.
In the following sections, we will introduce each dataset in more detail and explain how we used them.

Spike Dataset This dataset consists of multivariate time series with $D = 3$ features generated using a non-linear auto-regressive moving average (NARMA) process. Sudden spikes are introduced randomly in all features, but only the first feature is relevant and influencing the output label. The binary outcome y_t changes from 0 to 1 when a spike occurs in the first feature for the first time. In order to make this dataset suitable for a classification problem, we take the outcome y_t at the last time step as the class label. Therefore, our version of the spike dataset is a binary classification dataset (with labels 0 and 1) where whenever at least one spike occurs in the feature 0 the class will be 1 and otherwise it is 0.

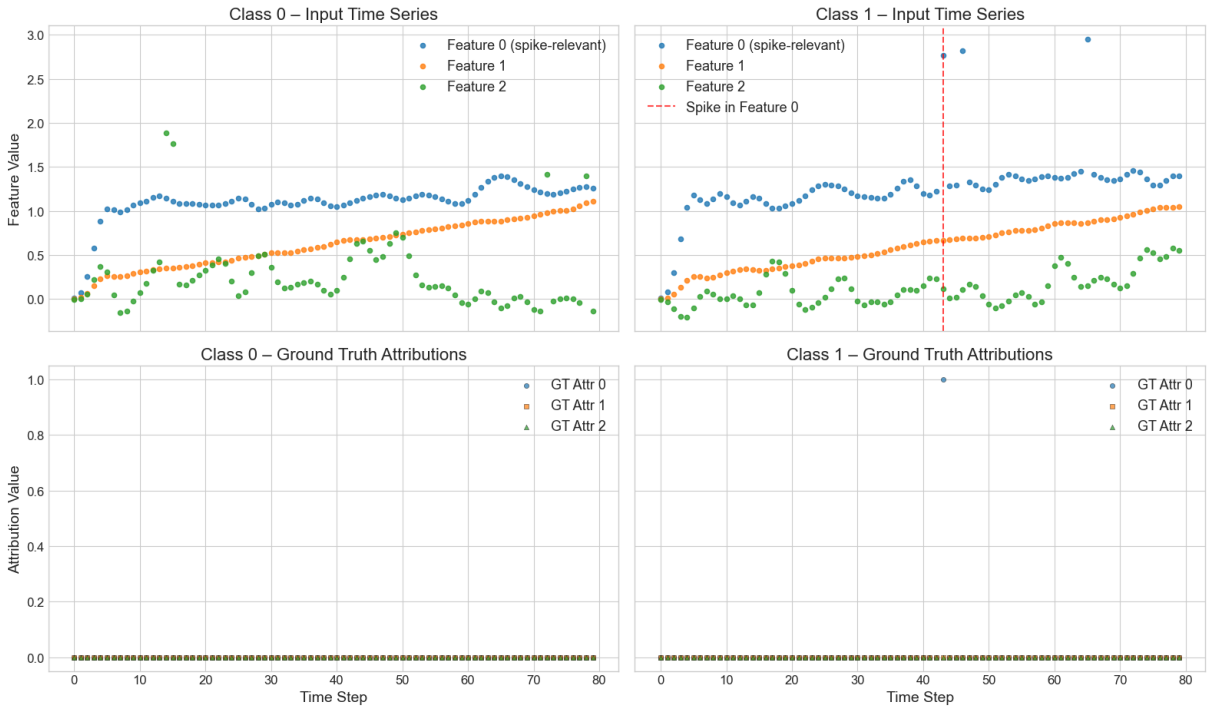

Figure 3: Example input samples and ground truth attributions from the spike dataset.

We use this dataset as a controlled test to evaluate the performance of the attribution methods we used. Since it is easy to identify the important feature and the precise time step, it is also easy for us to evaluate the natural language explanation. We generated these data using the code available from [30] with 1000 samples, 800 for the train, and 200 for the test set. Each sample has 3 features and 80 time steps.
Figure 3 illustrates two samples. In Class 1, a visible spike in feature 0 triggers the class switch, and the corresponding ground truth attribution reflects this. In contrast, Class 0 contains no such event, and all attributions remain zero.

UCI HAR dataset The UCI Human Activity Recognition (HAR) dataset is a widely used real-world dataset for time series classification [39]. It contains sensor readings collected from a smartphone worn on the waist of 30 participants performing six daily activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying. During the experiments, each participant wore a smartphone on their waist, which recorded data using its embedded accelerometer and gyroscope sensors. We chose this dataset because its activities are simple and easy to understand from the input features using some basic logic and knowledge of physics. This makes it suitable for evaluating the correctness, faithfulness, and quality of generated natural language explanations from our methodology.
The dataset includes raw triaxial signals, providing nine input features that represent total acceleration, body acceleration, and angular velocity along the x, y, and z axes. These signals are not normalized. The unit for acceleration signals is in gravity units (g) and for angular velocity is in $radians/second$. The dataset also includes features extracted from these nine raw signals, but we do not use those engineered features in this work. Each sample data is represented as a fixed-size window of 128 timesteps (2.56 seconds), sampled at 50 Hz.
Figure 4 illustrates one representative sample from each of the six activity classes in the UCI

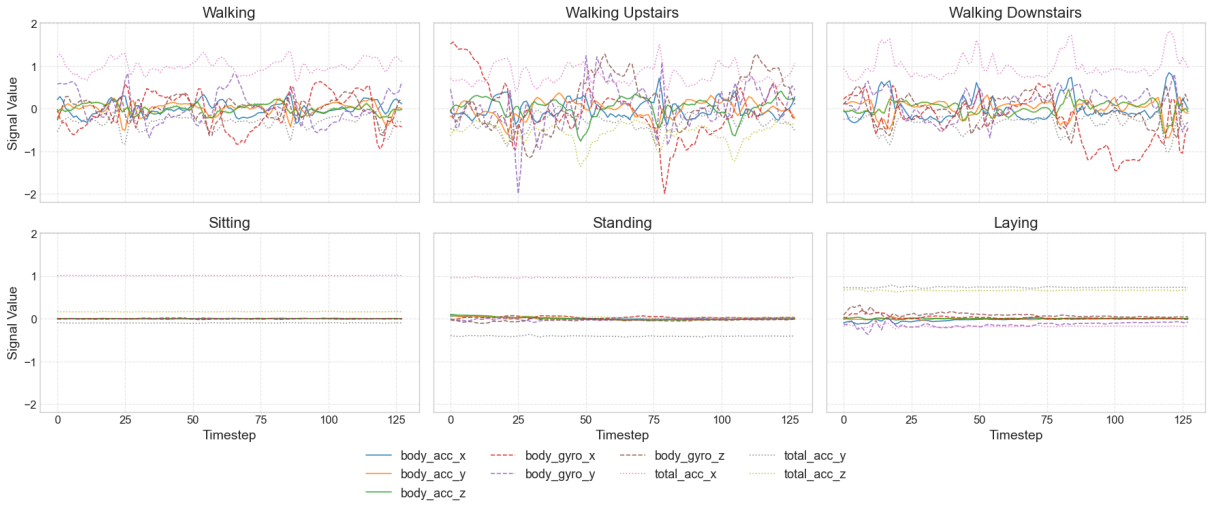

Figure 4: Example input samples from each activity class in the UCI HAR dataset.

Table 1: Number of samples per activity in the UCI HAR dataset

Activity	**Train Samples**	**Test Samples**
WALKING	1226	496
WALKING_UPSTAIRS	1073	471
WALKING_DOWNSTAIRS	986	420
SITTING	1286	491
STANDING	1374	532
LAYING	1407	537

HAR dataset. Each sample shows the raw triaxial signals (accelerations and angular velocities). These signals form the input to our model and are used without any engineered features.
The dataset was randomly partitioned into two sets: 70% of the volunteers were selected for generating the training data, and 30% for the test data. The dataset has 7352 training sample and 2947 test samples and is well-balanced across the six activity classes in both the training and test sets. Table 1 shows the number of samples per activity in each set.

3.2 Classification Models

This section describes the training and evaluation setup for a deep neural network time series classification model. In order to identify the best hyperparameters and prevent overfitting, we perform a hyperparameter optimization (HPO) and select the best hyperparameter set based on the models' performance on the validation dataset. A K-fold cross-validation method is implemented with $k = 5$. The tunable hyperparameters include the batch size, learning rate, number of epochs and model-specific architectural hyperparameters.
The deep classification models have been implemented in PyTorch, and HPO is conducted using the Optuna optimization framework. We use the Adam optimizer and cross-entropy loss for training the models. To avoid overfitting we apply an early stopping strategy. The training process is stopped if the validation performance does not improve for a fixed number of consecutive epochs. In our experiments, we set the early stopping patience to 5 epochs. To ensure reproducibility, we fixed random seeds during all training and evaluation phases. All

training and evaluation experiments were executed on a T4 GPU on google colab.
We employ two deep learning-based architectures for time series classification with the following specifications:

- **Recurrent Neural Network (RNN) Based Classifier**: The RNN model implemented for this experiment is designed using simple recurrent layers. The overall architecture consists of the following components:

  - A recurrent layer that can be either a GRU or LSTM, depending on the selected configuration. This layer processes the input sequence and encodes temporal dependencies in a hidden state.
  - An optional bidirectional setting that allows the model to read the time series in both forward and backward directions to capture future context in the time series.
  - A multi layer perceptron (MLP), which takes the final hidden state and outputs class logits (performs final classification).

  The architectural hyperparameters of this model include the hidden dimensionality of the recurrent units ($hidden\_dim \in \{16, 32, 64, 128\}$), the number of recurrent layers ($num\_layers \in \{1, 2, 3\}$), the type of recurrent unit used ($rnn\_type \in \{\text{GRU}, \text{LSTM}\}$), and whether the model is bidirectional ($bidirectional \in \{\text{True}, \text{False}\}$).

- **Transformer-Based Classifier**: The Transformer model implemented for this experiment is designed to use simple transformer encoder layers. The overall architecture consists of the following components:

  - A fully connected layer that embeds the raw time series data into a higher-dimensional embedding space.
  - A learnable positional encoding layer that provides information about the position of each time step in timeseries.
  - Transformer encoder layers that composed of multiple self-attention layers to allow the model to focus on relevant parts of the time series while ignoring irrelevant patterns. Each layer consists of a multi-head self-attention, feed forward and layer normalization and dropout.
  - A fully-connected MLP, that analyzes the extracted features and performs as the final classification head.

  The model includes many tunable architectural hyperparameters that are optimized during the HPO process. These include the hidden dimensionality of the transformer layers ($d\_model \in \{32, 64, 128\}$), the number of attention heads ($nhead \in \{2, 4, 8\}$), the number of transformer encoder layers ($num\_layers \in 1, 2, 3$), and the size of the MLP hidden layer ($hidden\_dim \in \{16, 32, 64\}$).

3.2.1 Classification evaluation

During the HPO process, we evaluate model performance on the validation set and select the hyperparameter configuration with the highest average validation accuracy. After the HPO process, using the best hyperparameter configuration, we retrain the model on the entire training set. Then we perform a final evaluation on the best model using the test dataset, reporting accuracy, precision, recall, F1-score to report the final performance metrics.

3.3 Attribution methods

To interpret the predictions of the trained classification models, we apply multiple post hoc point-based attribution explanation methods. These methods generate an attribution score showing the importance or influence of each time step in each feature on the model's output. We should note that we focus on point-based methods because they provide fine-grained attribution values that are well-suited for our goal of generating detailed and helpful NLEs. Other types of attribution-based methods, such as subsequence-based or instance-based explanation methods, often produce outputs that are already more interpretable and may not require NLEs to the same extent.

The attribution methods are implemented using the `tint` library. We experimented with several attribution techniques, including Temporal Integrated Gradients [29], Temporal Occlusion [30], DynaMask [31], and ExtremalMask [32]. We apply the attribution methods to the trained model checkpoints on samples from the test dataset. Attribution values are computed for each input sample per feature per time step. The generated attribution maps are then stored to be used in order to generate text explanation. Depending on the method, different parameters, such as baselines or input shapes, are configured.

3.3.1 Attribution-based explainer evaluation

To assess the quality of explanations generated by attribution methods, we use Area Under the Receiver Operating Characteristic Curve (AUROC) and Area Under the Precision-Recall Curve (AUPRC) metrics calculated using the ground truth labels for each time step.

AUROC. AUROC measures how well the explainer ranks truly important time steps above unimportant ones. Shows the probability that a randomly chosen important time step receives a higher attribution score than a randomly chosen unimportant one. An AUROC of 1.0 indicates perfect separation, while 0.5 indicates random guessing.

$$\text{AUROC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(x))\, dx$$

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

AUPRC. AUPRC focuses on the explainer's precision-recall tradeoff, which is particularly relevant for the Spike dataset, where most time steps are unimportant and only a single truly important time step exists in each positive (class 1) sample.

$$\text{AUPRC} = \int_0^1 \text{Precision}(r)\, d\text{Recall}(r)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

We computed AUROC and AUPRC using `sklearn.metrics.roc_auc_score` and `sklearn.metrics.average_precision_score` respectively, applied to flattened attribution scores and binary ground truth labels.

We should note that evaluating the quality of this form of explanation is only possible on the spike dataset since we know the ground truth attribution values. In addition, the classification

models on the spike dataset perform with near-perfect accuracy (close to 100%), which also makes it possible to evaluate attribution-based explainers in this way, despite the black-box nature of the models.
For experiments on the UCI HAR dataset, we cannot quantitatively evaluate the attribution scores due to the lack of ground-truth importance labels. Instead, we rely on the best performing methods from the Spike dataset and use those as our attribution-based method.

3.4 Categorizing attribution values

Attribution-based explanation methods assign an importance score to each input feature or time step, indicating its contribution to the model's prediction. In the attribution-based methods applied in this study, as discussed in Section 2.3, higher attribution values suggest a greater influence on the classifier's decision-making process. To generate explanations, we needed a way to identify and prioritize the most influential parts of the input time series based on these scores. To do this, we aggregate the absolute attribution values across all test samples, time steps, and classes, forming a single distribution of absolute attribution values. Our goal was to classify and categorize values of the most influential points that could guide NLE generation. Initially, we experimented with common thresholding strategies, such as selecting setting manual cutoffs by inspecting the distribution, or using fixed percentiles. However, a fixed global threshold can result in too many or too few points being marked as influential depending on the overall attribution of the sample. Also, these approaches required manual tuning across different datasets which made them unreliable and not general enough. To improve robustness and generality, we decided to categorize the attribution values into three discrete influence levels. We tried binning methods using linear and exponential relations, but they still relied on the distribution shape and didn't generalize well.
In the end, we chose a clustering-based approach using the K-means algorithm. We ran K-means with $k = 3$ on the full distribution of attribution values to find natural boundaries between the influence levels. This method does not need manual adjustment and is automatically adjusted to the value distribution, which makes it more robust across datasets and attribution methods. We implemented this using the `KMeans` class from the `sklearn.cluster` library.

3.5 Natural language explanation generation

In the final stage of our pipeline, we translate attribution values into human-readable explanations through two strategies: a rule-based template approach and a prompt-based LLM approach. For each dataset, we choose the best performing classification model with the best performing attribution method, and use their output to generate the NLE.
Before generating explanations, we needed a way to filter out the most influential features and time steps based on the attribution values. This preprocessing step was necessary for the rule-based method and was used as structured guidance in some prompts in the prompt formulation for LLM-based explanations. We categorized attribution scores into three levels: strong, moderate, and none.
In the following subsections, we first describe the metrics used for evaluating the quality of NLEs. We then provide a detailed explanation of categorizing attribution values process, followed by the explanation generation methods: first the rule-based template approach, and

then the prompt-based LLM approach. Finally, we present our evaluation setup for assessing both forms of NLEs across the two datasets.

3.5.1 NLE evaluation metrics

To guide the design of both template-based and LLM-based explanation methods we used the following set of evaluation metrics. These metrics later used in our final user evaluation.

- **Faithfulness:** This metric measures if the explanation highlights the correct features and time steps that contributed most to the prediction. A faithful explanation should clearly distinguish important from unimportant inputs and be consistent with the attribution values in our problem.
- **Completeness:** Completeness evaluates whether the explanation covers all important contributing regions. An incomplete explanation may mention only a subset of influential points.
- **Clarity:** For an explanation to have clarity, it must be easy to understand. A clear explanation uses accessible language and structure, making it easy for the user to follow, regardless of their technical background.
- **Helpfulness / Usefulness:** This metric measures whether the explanation helps the user understand why the model made a particular prediction.

To define the thresholds between these levels, we sort the cluster centers in ascending order and compute the midpoints between each pair. More formally, given the sorted cluster centers $\{c_1, c_2, c_3\}$, we calculate the thresholds as:

$$\text{thresholds} = \left[\frac{c_1 + c_2}{2}, \frac{c_2 + c_3}{2}\right]$$

These two threshold values divide the attribution scores into three bins, which are then used to assign each time step to one of the three influence categories for NLE generation.

3.5.2 Template based NLE

We designed two different output formats, each designed for a specific end user: either a non-expert user or an expert user. For each type of user, there are two variations, depending on whether any influential data points were identified in the input or not.
We categorize each feature-time step combination into categories of influence: no influence, moderately influential, and strongly influential as described in 3.4. Then, to determine the most influential features, we ranked them based on the weighted count of moderately and strongly influential points, giving a higher weight to strong influence. Moreover, we only considered features as most influential if they included at least one strongly influential time step. This helped with keeping the explanation concise.

Non-expert Template: For non-expert users, we use a simplified explanation that highlights at most three most influential features and up to three time steps per feature.

With influential points:

The model predicted this input as class <predicted_label>. The most important parts influencing the model's decision were:

- Feature <F0> at time steps:
  - <T1> (<positive/negative> influence)
  - <T2> (<positive/negative> influence)
- Feature <F2> at time steps:
  - <T20> (<positive/negative> influence)

Positive influence means that these inputs helped the model choose <predicted_label>, while negative influence means that they pushed it away from that choice.

With no influential points:

The model predicted this input as class <predicted_label>. No specific part of the input was important in the model's decision-making process, and the decision was not based on any specific feature or time step.

Expert Template: For expert users, the explanation includes all points identified as strongly and up to 5 timesteps of each moderately influential feature, along with their attribution values and influence direction.
With influential points:

The classifier predicted this sample as class <predicted_label>. Attribution scores were computed using <attribution_method>, and clustered into three influence levels using k-means.
The most influential points are:
Strongly Influential

- Feature <F0> at time steps:
  - <T1> (<positive/negative> influence, attribution = <value>)
  - <T2> (<positive/negative>, attribution = <value>)

Moderately Influential

- Feature <F3> at time step:

- <T4> (<positive/negative> influence, attribution = <value>)

- Feature <F1> at time step:
  - <T5> (<positive/negative> influence, attribution = <value>)

Other features and time steps had minimal influence based on the attribution method.

With no influential points:

The classifier predicted this sample as class <predicted_label>. Attribution scores were computed using <attribution_method> and clustered into three influence levels using k-means. Attribution scores were analyzed, but no feature or time step reached the moderate or strong influence thresholds. This suggests that the model made its decision without relying on any specific part of the input.

3.5.3 LLM based NLE

In this approach, we treat NLE generation as an in-context learning problem, where we prompt an LLM with structured information about prediction, attribution values, and some other different context and ask it to generate a natural language explanation.
We use a pre-trained LLM to generate natural language explanations based on the attribution values. We use the `llama-3-70b-instruct` model via the `replicate` API, which provides access to llms running on a hosted infrastructure. The inference setup uses a fixed random seed for reproducibility (`seed=42`) and a temperature of 0 to ensure deterministic output. In our explainability task we don't need the model to be creative so we concluded that setting the temperature to 0 won't have a negative influence on the quality of the explanation. In our experiments,, the output of the LLM was mostly consistent and reproducible with these settings.
An important step here was the prompt engineering process. We needed the NLE to be faithful and reliable according to the given attribution values. It was also important to us that the final textual explanation includes all the important features and time steps and explains the classification decision process clearly for the end user.
In order to find the best prompt according to these criteria, we tried different wording, different ways of passing the attribution numbers, and different instructions. To improve the quality of the explanations, we iteratively changed and refined the prompt template based on the quality of the generated outputs. Since the Spike and UCI HAR datasets represent two very different problem contexts, the prompt engineering process also differed between them. The Spike dataset, used as a pre-study, allowed us to explore and identify effective strategies for passing attribution values and for including (or excluding) specific information. These lessons guide our approach for the HAR dataset, where we tried to improve the prompt and NLEs by adding context about the real-world dataset. We added this context to generate deeper and more informative NLEs.

Prompt Engineering for the Spike Dataset In the spike data set, we experimented with different variations on one sample from each class using attribution values resulting from

the temporal integrated gradients attribution method. We started with a general template that described the prediction of the model and includes the attribution values resulting. The initial template (#1) was as follows:

```
A deep time series classifier has been trained on labeled data. After
training, an attribution-based explanation method was applied to
compute attribution values for each time step of each feature of a
specific input sample.
The attribution scores are as follows: <attributions>
The classifier predicted the sample to belong to class
<predicted_label>.
Explain the classifier's prediction for this input. Focus on the most
influential features and time steps based on the attribution scores.
Keep the explanation concise.
```

Initially we provided the `<attributions>` as an array of scientific notation numbers, with the same length as the input time series. However, we found out that with this number formatting the LLM is having trouble comparing and analyzing values so we round the attribution with 2 decimals and formatted values in a normal way for the rest of the prompt engineering process. This helped the LLM to identify the most influential attribution values better, but it was still struggling with referencing the exact time steps, often generating vague time intervals as influential time steps (i.e., Time steps 35–39), while there exist only one important time step. In order to solve this issue, we restructured the attribution input to present the values per feature and per time step, using a dictionary format. Each feature was printed separately with time step-to-value mappings, such as `For feature 0: 0: 0.000, 1: 0.001, ..., 79: 0.000` in template (#2). This helped the model consider each time steps and made the input more readable. At this stage, the generated explanation was accurate and acceptable for samples from class 1. However, the LLM struggled with class 0 samples, where it did not recognize that all attribution values were uninfluential. In class 0 samples, most values were 0 and others very close to zero (typically below 0.01). According to domain experts, such small attribution values should not be interpreted as significant, especially given the potentially wide value range of the attribution method. But the LLM tended to treat these minimal values as meaningful, probably because it was still trying to identify and explain influential points even when none were truly present.

To solve this problem, we tried different approaches to give the LLM more context about the characteristics of influential points. Table 2 shows each prompt variation (all adding context by modifying prompt #2). Including a description of the change and its impact on the model's output. The full text of each prompt version used during experimentation is provided in Appendix A.

From the qualitative results in Table 2, we chose prompt #6 and prompt #7 for generating final explanations and used them in the final user evaluation step.

Prompt Engineering for the UCI HAR Dataset For the UCI HAR dataset, we followed a similar prompt engineering process to the one used for the Spike dataset, but adapted our approach to handle the higher input dimensionality, longer sequences (128 time steps), and more complex sensor data. Due to the large number of signals and the context

length limits of the LLM, we first applied a threshold to filter out attribution values below 0.01 to reduce prompt size and improve clarity. The initial prompt (Prompt #1) was as follows:

```
A deep time series classifier has been trained on labeled data. After
training, an attribution-based explanation method was applied to
compute attribution values for each time step of each feature of a
specific input sample.
The attribution scores are as follows (formatted as time_step:
attribution_value for each feature):
<attributions>
The classifier predicted the sample to belong to class
<predicted_label>.
Explain the classifier's prediction for this input. Focus on the most
influential features and time steps based on the attribution scores.
Keep the explanation concise.
```

In this prompt, attributions were formatted per signal and per time step. The placeholder <attribution_dict> represents a per-feature mapping of time steps to attribution values.
While this formatting produced good explanations for samples from different classes, we observed that the LLM occasionally didn't mention most influential attribution points and sometimes hallucinated additional feature importance. To address these issues, we explored several prompt variations. Table 3 summarizes the prompt engineering strategies we evaluated and their observed effects on explanation quality. The full text of each prompt is provided in Appendix A.
Among these, Prompt #3, which combined attribution filtering with contextual dataset information, resulted in the mostly faithful, complete, and interesting explanations. However, even with this improved structure, we noticed that the model sometimes hallucinated and tried to use its general knowledge to infer human motion patterns from the attribution values. It seemed that the LLM is treating attribution values as if they were raw input signals. This is not a correct reasoning process, since attribution values only indicate importance, not the actual content of the signal.
To reduce this kind of hallucinations, we introduced Prompt #4, which builds on Prompt #3 by adding a simple instruction: *Do NOT infer sensor magnitudes or human motions. Your explanation must mention importance, not what happened in the signal.* This constraint helped the model stay focused on attribution-based reasoning and prevented unsupported inferences about the input signal.
We therefore selected this prompt as our final choice for generating explanations in the user evaluation.

Prompt Adaptation Based on User Type: After selecting the most effective general prompt for each dataset, we customized the final prompt based on the intended end user. The core content and structure of the prompt remained the same, but the instruction was changed to specify the end users' need to improve the usefulness.
For the Spike dataset:

- **Expert user:** `"Explain the classifier's prediction for this sample to an expert user. Keep the explanation concise and mention which attribution`

`method was used."`
We also included the name of the attribution method directly in the prompt context.

- **Non-expert user:** `"Explain the classifier's prediction for this sample to a non-expert user. Keep the explanation concise."`

For the UCI HAR dataset:

- **Expert user:** `"Explain the classifier's prediction for this sample to an expert user, using the attribution scores and mentioning influential features and time steps. Keep the explanation concise."`
We also included the name of the attribution method directly in the prompt context.

- **Non-expert user:** `"Explain the classifier's prediction for this sample to a non-expert user. Keep the explanation short and concise."` To improve usefulness and help the model generate clear and concise responses for non-expert users, we also included two examples of short, informative explanations in the prompt.

3.6 NLE Evaluation

The quality of the generated NLEs was evaluated by human judgment.
For the Spike dataset, evaluation was conducted internally by the authors. We selected the two best-performing prompts from our prompt engineering process, Prompt #6 and Prompt #7 (see Table 2) and applied them to a set of 20 samples (10 randomly selected from each class). Each explanation was manually reviewed and rated according to three core criteria: faithfulness, clarity, and completeness. Ratings were assigned using a 5-point Likert scale and an additional overall satisfaction score was recorded for each explanation.
For the UCI HAR dataset, we conducted a user study using a custom web-based survey platform for both expert and non-expert users.[1] Each evaluator was shown both a template-based and an LLM-based textual explanation, each accompanied by plots. Then they were asked to rate them based on the 4 metrics: faithfulness, helpfulness, completeness, and clarity (previously defined in Section 3.5.1). Each participant evaluated 12 samples, 2 from each of the 6 classes in the HAR dataset. From these, 6 samples were fixed across all users to enable controlled comparison, while the remaining 6 were randomly selected from the test set to increase variability. Users who self-identified as familiar with XAI (rating 4 or 5 on a 5-point scale) were considered experts and shown explanations tailored for expert users. Participants evaluated each explanation independently using a 5-point Likert scale across the defined metrics. In addition the defined metrics, they answered a modality preference question indicating which form of explanation they preferred in that context (text only, plots only, both, or neither).
In addition, the survey included two overall questions per sample: one asking which textual explanation style (template vs. LLM) they preferred, and another asking which modality (text, plots, both, or neither) best helped them understand the model's decision. These questions allowed us to also assess user preference and usefulness of different NLE styles and visualization combinations for explaining a AI model through attribution values.
The evaluation interface and participant instructions are shown in Appendix C and B, respectively.

[1] `https://tsnle.vercel.app`

Table 2: Prompt Strategies and Observed Effects on Explanation Quality (Spike Dataset)

Strategy	Prompt Variant	Observed Effects
Attribution Categorization	Prompt #3	Included the name of the attribution method, the method's upper and lower bounds, and the relation of attribution values to influence. These additions did not improve the explanation on class 0. The model still treated very near-zero values as important.
Top-k Attribution Forcing	Prompt #4	Instructed the model to list only the top 3 most influential feature-time step combinations. The model forced itself to generate exactly three influential points, even when none were meaningful. This was especially problematic in class 0 cases and overall even reduced explanation quality on the class 1 sample.
Attribution Categorization	Prompt #5	Described thresholds for no influence, slightly influential, and strongly influential attribution values (using K-means clustering as described in Section 3.4). This improved explanation faithfulness and the model correctly mentioned the lack of strongly influential features on the class 0 sample. However, it still sometimes mentioned "slightly influential" points that were not actually present in the input.
Attribution Categorization (Binary)	Prompt #6	Defined a threshold for influence based on the second bin center from the K-means clustering (dividing points into influential / not influential). The results improved for class 0, as the LLM correctly stated that "no influential features or time steps were detected," and explained the prediction as not driven by any specific point. However, the explanations were slightly less detailed for class 1 samples, since this approach used only two influence levels.
Attribution Filtering	Prompt #7	Instead of passing the full attribution array, only moderately and strongly influential feature-time step combinations were passed, based on K-means bin labels. The prompt included a list of these points, or stated that none were found. Explanations were accurate, faithful, and clear. Since only influential points were passed, the LLM no longer needed to identify them itself.

Table 3: Prompt Strategies and Observed Effects on Explanation Quality (UCI HAR)

Strategy	Prompt Variant	Observed Effects
Base Formatting	Prompt #1	Clean structure with per-feature attribution dictionary. The LLM handled the format reasonably well but occasionally ignored key values or hallucinated influential features.
Attribution Filtering	Prompt #2	Passed only moderately and strongly influential feature-time step combinations. Explanation quality improved and was mostly faithful and complete across samples. But we wanted to see how explanations reasoning and storytelling improve when the LLM have context about the dataset.
Dataset Context Injection	Prompt #3	Adding sensor details (e.g., placement and axis meanings) helped the LLM explain using real-world reasoning. Prompt #3 was usually complete, faithful, and interesting. However, in some cases the LLM confused attribution values with input sensor values and interpret attribution values in a way to justify the classifiers prediction in an incorrect way.
Attribution-Only Constraint	Prompt #4	Added an explicit instruction to avoid inferring attribution values as input motion signal. This eliminated hallucinations and improved faithfulness in explanations. Selected as the final prompt for evaluation.
Input Value Context	Prompt #5	Combined #4 with actual input values of most influential features. Resulted in generic, repetitive explanations and less attention to attribution values during reasoning.

4 Results

In this section, we present the results of our experiments on the Spike dataset and the UCI HAR dataset. These results include model performance on validation and test sets, as well as the effectiveness of the attribution methods, and evaluating natural language explanations. Each data set is discussed separately, starting with the Spike data set as a baseline, followed by the more complex real-world UCI HAR dataset.

4.1 Spike dataset (Baseline)

Model Performance: The best performing configuration for the Transformer model was the following:

- `batch_size`: 32, `learning_rate`: 0.0093, `epochs`: 20
- `d_model`: 64, `nhead`: 8, `num_layers`: 1, `hidden_dim`: 64

The best performing configuration for the RNN model was:

- `batch_size`: 32, `learning_rate`: 0.0055, `epochs`: 11
- `hidden_dim`: 128, `num_layers`: 2, `rnn_type`: GRU, `bidirectional`: False

Table 4 presents the validation accuracy and final test evaluation metrics for both models.

Table 4: Performance results of Transformer and RNN classifiers on the Spike dataset.

Model	Val. Acc	Test Acc	Precision	Recall	F1-score
Transformer	0.998	0.995	0.996	0.991	0.994
RNN (GRU)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

These results show that both models perform really well and closely on the test dataset, indicating that they have been trained almost perfectly. For our pipeline, this means that we can be confident that the model is paying attention to the right features, and we can reliably evaluate the attribution methods using the ground truth attribution values in this dataset.

Attribution Analysis: For each sample in the Spike dataset, each timestamp in each feature has a ground truth attribution value of 0 or 1. This allows us to perform per-timestep evaluation on the generated attribution values. Formulating the problem as if each timestamp (across all channels) is a binary classification therefore, we can calculate AUROC and AUPRC over all samples in the test dataset.
Table 5 shows AUROC and AUPRC for the different attribution techniques we used, evaluated on both the Transformer and the RNN models. To assess the robustness of each attribution method, we divided the test set into 5 folds and report the mean and standard deviation of AUROC and AUPRC scores across these folds.

Table 5: Per-timestep AUROC and AUPRC scores for different attribution methods on the Spike dataset.

Attribution Method	Classifier	AUROC	AUPRC
Temporal Integrated Gradients	Transformer	0.996 ± 0.007	$\mathbf{0.963 \pm 0.032}$
	RNN	$\mathbf{0.998 \pm 0.001}$	0.791 ± 0.042
Temporal Occlusion	Transformer	0.992 ± 0.013	0.922 ± 0.057
	RNN	$\mathbf{0.998 \pm 0.001}$	0.785 ± 0.043
DynaMask	Transformer	0.670 ± 0.101	0.005 ± 0.002
	RNN	0.627 ± 0.017	0.002 ± 0.001
ExtremalMask	Transformer	0.627 ± 0.149	0.022 ± 0.026
	RNN	0.907 ± 0.040	0.011 ± 0.007

Based on the results in Table 5, the Integrated Temporal Gradients had the best overall performance. It achieved both high AUROC and high AUPRC across models. Temporal Occlusion also performed well and was close behind. In contrast, some methods, like ExtremalMask for RNN, had a high AUROC but a very low AUPRC. This means that the model was able to identify the correct important points, but it still had many false positives.
DynaMask and ExtremalMask both perform much worse compared to the other methods. One reason is that they are not gradient-based. They work by masking parts of the input and checking how the model's output changes, but this approach is more suitable for image data where important regions are usually larger and continuous.
Because Temporal Integrated Gradients gave the best performance when combined with the Transformer model, we chose the attribution values from this combination to generate the natural language explanations

NLE Evaluation: We randomly selected 10 samples from each class (class 0 and class 1) and generated explanations using each prompt. The generated outputs were manually evaluated on the basis of faithfulness, completeness, clarity, and overall satisfaction.
We evaluated each explanation, using a five-point Likert scale (1 = poor, 5 = excellent). Table 6 reports the average evaluation scores for prompt #6 and prompt #7 for spike dataset. We include the standard deviation to reflect the consistency of each prompt across different samples. If the standard deviation is high it shows that the quality differed substantially from one sample to the next.

Table 6: Mean and standard deviation of evaluation scores for Prompt #6 and #7.

Prompt #	Faithfulness	Completeness	Clarity	Overall Satisfaction
6	3.80 ± 2.07	4.85 ± 0.36	5.00 ± 0.00	4.10 ± 1.29
7	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00

While Prompt #7 achieved perfect scores and had the best performance, Prompt #6 showed lower mean faithfulness and completeness scores and higher variability. Prompt #7 performed significantly better than Prompt #6 in Faithfulness, Completeness, and Overall Satisfaction (Sign Test, $p < 0.05$), with no significant difference in Clarity. Prompt #6 showed lower mean

faithfulness and completeness scores and higher variability. To understand this variation, we break down the results of Prompt #6 by class in Table 7.

Table 7: Per-class mean and standard deviation of evaluation scores for Prompt #6.

Class	Faithfulness	Completeness	Clarity	Overall Satisfaction
0	2.60 ± 2.42	4.70 ± 0.48	5.00 ± 0.00	3.20 ± 1.32
1	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00

Prompt #6 performed very well on class 1 samples, achieving perfect scores across all evaluation metrics. However, its performance decreased significantly when applied to samples in samples in class 0 samples. Specifically, scores for faithfulness and overall satisfaction dropped considerably. This can be attributed to the characteristics of the class 0 examples, where all the attribution values are either zero or very close to zero, making it harder for the LLM to correctly reason about importance.
We observed several recurring issues in the generated explanations for class 0. In some cases, the LLM incorrectly marked near-zero attribution values as being above the stated threshold (1.126), even when they were not. In other examples, the LLM correctly stated that no influential points were present but still went on to describe certain time steps as having contributed positively, contradicting the attribution input. Additionally, when higher attribution values did exist, the model sometimes failed to mention the most significant points, instead highlighting less relevant ones. These inconsistencies led to lower faithfulness and overall satisfaction scores in the explanations provided for class 0 samples.
In contrast, Prompt #7 avoids these issues by providing only the filtered set of influential points, leading to consistently faithful and clear explanations. This justifies our final choice of prompt #7 as the most robust and reliable prompt for generating natural language explanations in both classes.

4.2 UCI HAR dataset

4.2.1 Model Performance

The best performing configuration for the Transformer model was the following:

- `batch_size`: 32, `learning_rate`: 0.0011, `epochs`: 36
- `d_model`: 64, `nhead`: 8, `num_layers`: 3, `hidden_dim`: 32

The best performing configuration for the RNN model was:

- `batch_size`: 32, `learning_rate`: 0.0062, `epochs`: 29
- `hidden_dim`: 64, `num_layers`: 2, `rnn_type`: GRU, `bidirectional`: True

Table 8 presents the validation accuracy and final test evaluation metrics for both models per class and overall. The RNN performed better or similar to the Transformer in most classes, especially in dynamic activities like WALKING. While the Transformer did slightly better on the LAYING class, the RNN had more balanced results overall, showing that it is more reliable across different activities. The RNN consistently outperformed the Transformer in all micro and

Table 8: Per-class test metrics for Transformer and RNN classifiers on the UCI HAR dataset. Macro, micro, and accuracy values are included in the last rows.

Class	Transformer			RNN (GRU)		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
WALKING	0.966	0.869	0.915	0.981	0.942	0.961
WALKING_UPSTAIRS	0.926	0.926	0.926	0.927	0.975	0.950
WALKING_DOWNSTAIRS	0.875	0.983	0.926	0.926	0.979	0.951
SITTING	0.776	0.863	0.817	0.790	0.798	0.794
STANDING	0.870	0.771	0.817	0.814	0.805	0.809
LAYING	0.993	1.000	0.996	1.000	0.950	0.974
Macro Avg.	0.901	0.902	0.900	**0.906**	**0.908**	**0.907**
Micro Avg.	0.900	0.900	0.900	**0.905**	**0.905**	**0.905**
Test Accuracy		0.900			**0.905**	
Validation Accuracy		0.942			**0.982**	

Table 9: Average self-reported XAI familiarity, AI usage frequency, and AI trust levels of participants.

Expertise	**XAI Familiarity**	**AI Usage**	**AI Trust**	**Count**
Expert	4.33 ± 0.58	5.00 ± 0.00	3.33 ± 1.15	3
Non-Expert	2.11 ± 0.93	4.33 ± 0.71	3.22 ± 0.97	9

macro metrics reported. We chose the RNN model to apply attribution method and generate NLEs for samples of the HAR dataset.
We should note that the goal of this work is not to achieve state-of-the-art performance on the UCI HAR dataset, but to develop fairly accurate classifiers suitable for explanation.
Since RNN shows better results on test dataset, we choose the RNN model to apply attribution-based explanation methods.

4.2.2 NLE Evaluation from user survey

A total of 12 participants (mean age = 28.17 years, range = 21–42) took part in the study. The majority held a master's degree (78%), while the remaining participants had a bachelor's degree, a high school diploma, or a Ph.D. Table 9 summarizes the average self-reported familiarity with XAI, AI usage, and trust levels by expertise.

NLE evaluation metrics: Figure 5 shows the mean user ratings for each explanation quality metric for both LLM-based and template-based NLEs, split by user expertise. In all metrics and for both user groups, LLM-based NLEs consistently receive higher average ratings than the template-based NLEs. The biggest differences are seen in helpfulness and clarity, especially the 3 participating expert users. This suggests that LLM-based explanations were generally more informative and easier to understand for both groups of users.
To analyze accurately we calculated p-values with a two-sided Sign Test, which compares the number of higher ratings for llm-based explanation samples (excluding ties) against the 50% expected by chance using a binomial model.

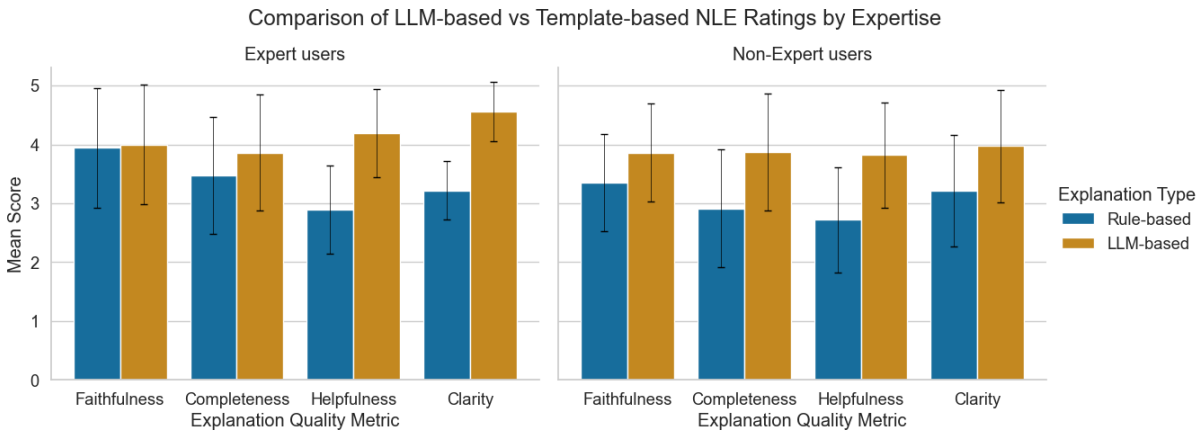

Figure 5: Comparison of LLM-based and Template-based NLE across quality metrics by expertise. Error bars are standard deviations over all samples of all users.

For non-expert participants, the differences in metric ratings were statistically significant for all four metrics: faithfulness ($p = 3.33 \times 10^{-5}$), helpfulness ($p = 5.92 \times 10^{-14}$), completeness ($p = 3.23 \times 10^{-12}$), and clarity ($p = 1.03 \times 10^{-7}$). Among expert participants, the largest and most significant differences were in helpfulness ($p = 8.05 \times 10^{-7}$) and clarity ($p = 5.65 \times 10^{-6}$), while differences in faithfulness ($p = 1.0$) and completeness ($p = 7.68 \times 10^{-2}$) were not statistically significant.

The ratings in *faithfulness* and *completeness*, were surprising to us since, as a fact, we know that the rule-based approach is fully faithful and mostly complete, but apparently the participants did not have this perception. Maybe the lesser clarity of these types of NLE made it harder for users to perceive these explanations as fully faithful. It is known that human evaluation metrics tend to be highly correlated, with participants often rating all aspects positively when they have an overall good experience, and all aspects negatively when their overall experience is poor, and vice versa.

Preferred textual explanation style: Figure 6 shows the percentage of responses to the question of which textual explanation style that, participants preferred on each sample split by expertise level.

Both expert and non-expert users showed a strong preference (77.8% of responses) for the LLM-based explanation style over the template-based alternative. This result confirms that users generally found the LLM-based NLEs more clear and understandable which is consistent with the earlier results of metrics ratings.

Only a small percentage of users preferred the template-based explanations: 6% of expert responses and 11% of non-expert responses. This suggests that rule-based, static explanations were rarely perceived as more helpful.

Overall, these results appear to indicate a clear user preference for LLM-generated textual explanations, but also suggest that a small group of users may view both approaches as similarly useful depending on the sample.

Modality Preference Shifts Between NLE Styles: To understand how the preferred form of explanation (plot or text) changes depending on the type of textual explanation, we analyzed participant responses to the explanation format preference question, which was asked

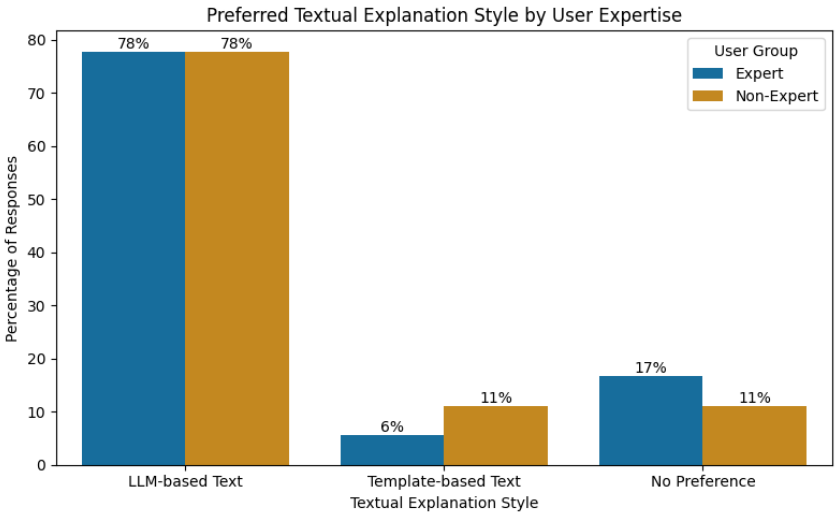

Figure 6: Preferred textual explanation style by user expertise. Each bar shows the percentage of responses selecting user preferred textual explanation style. (Experts: $n = 36$, Non-Experts: $n = 108$)

separately after showing each template-based and LLM-based NLEs on each sample.
Figure 7 presents a heatmap split by user expertise, showing the proportion and count of samples where users selected a specific form of explanation (text only, plots only, both, or neither) after reading each type of NLE.
The strongest shift in both expert and non-expert groups was from *plots only* under the template-based NLE toward *both* under the LLM-based NLE. This shift was particularly strong among non-experts (24%) and even more concentrated among experts (41%). On the other hand, the reverse transition was nonexistent for experts and very low for non-experts. This suggests that LLM-based NLE improved the perceived value of combining text and plots.
The second strongest shift for both groups was *both* to *both*. which suggest that in about 20% of cases, both textual explanation styles were seen as equally effective in combination with plots.
Another interesting trend appears for non-experts: around 29% of responses shifted from *neither, both, or plots only* to *text only* with the LLM NLE. This seems to indicate that for many samples, users found the LLM-based explanation strong enough to prefer just the text. A smaller but similar trend is visible for experts, with about 22% of responses shifting toward *text only* from other choices.
Together, these patterns may reflect that LLM-based NLEs increased users' confidence in textual information, encouraging a shift away from visual-only formats and, in some cases, even reducing the perceived need for plots. This effect is more pronounced for non-experts, hinting at the potential of well-structured and good natural language explanation to bridge gaps in technical familiarity and enhance interpretability.

Perceived helpfulness of explanation modalities: Figure 8 shows which explanation modality users found most helpful for understanding the model's decision making process. For both expert and non-expert users, the majority preferred having both textual and visual explanations together. This was the case for expert users 81% and for non-expert users 68% of the time. This seems to indicate that combining text and visualization modalities help both

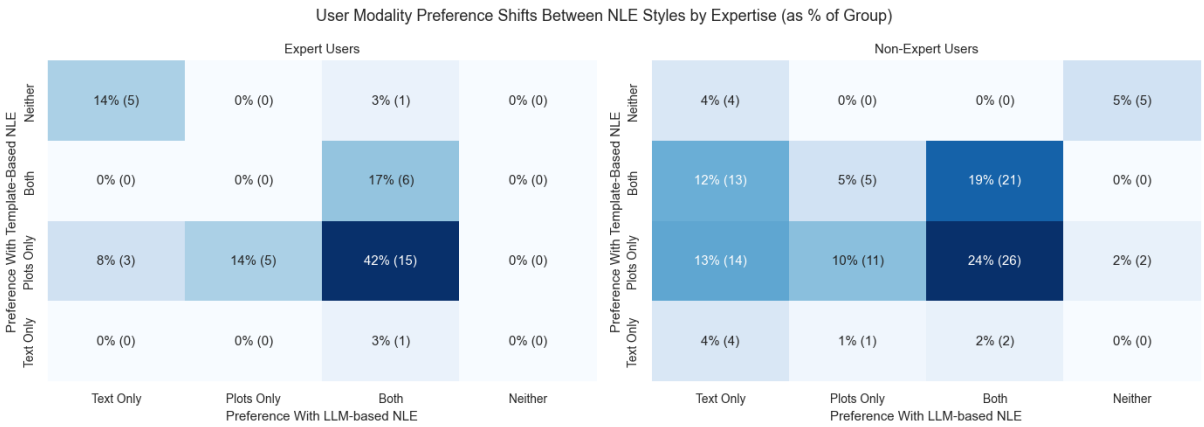

Figure 7: Modality preference shifts between template-based and LLM-based NLEs, split by user expertise. Each cell shows the percentage (and count) of samples where participants selected a specific explanation modality under each textual explanation type. (Experts: $n = 36$, Non-Experts: $n = 108$)

group of users to understand attribution results better.

Only for a small number of samples, users preferred using just one modality. Among non-experts, slightly more preferred *text only* (18%) than *plots only* (10%). The reason that *plots only* is a less helpful modality for non-experts may be due to their limited background knowledge of non-expert users on how they should interpret attribution values.

For experts, 3 times an expert user (each user once) preferred *plot only* and 3 times an expert user preferred *text only*. We observed that experts who preferred *plots only* did so only for samples from the LAYING, STANDING, and SITTING classes. For these classes, the attribution plots were generally simple, showing a few clearly influential time steps and features, which may remove the need for a textual explanation for an expert user.

It is also notable that the expert responses for both *text only* and *neither* came from the same participant, who later informed us that they had difficulty interpreting some of the complex attribution plots due to color blindness. This suggests that their choices may have been influenced more by accessibility issues than the quality of the explanations themselves.

Overall, these results are consistent with the idea that combined textual and visualization explanations are more helpful than using text or plots alone, and also highlight that sample attribution values complexity can influence which modality users prefer.

Qualitative feedback on textual explanations: At the end of the survey, participants were asked to describe what they found most and least helpful about the textual explanations. Although users were not told which explanation came from which system, we manually identified the references based on their descriptions.

Users found textual explanations helpful, some users mentioned that having *"specific reference to the points on the plots and the specific time steps made it easy to follow"*. One participant mentioned that the most helpful part of the textual explanations was *"mentioning what was the most important feature behind the classification label (and its range)"*.

For the template-based NLE, while the bullet-point format was easy to parse, one non-expert user felt that it was *"missing reference to relevant points"*. This may be due to the fact that the template-based NLE includes only the top 3 most influential features, keeping the explanation

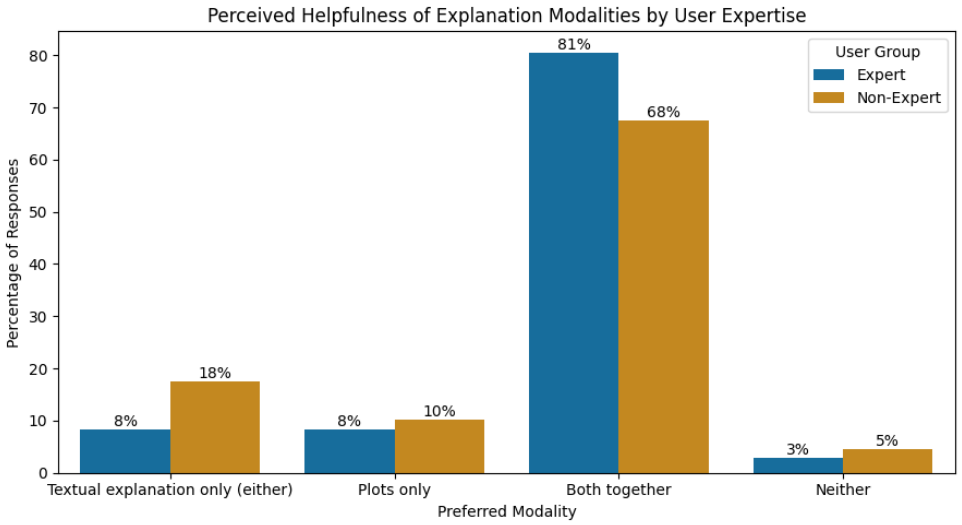

Figure 8: Perceived helpfulness of explanation modalities by user expertise. Bars show the percentage of responses selecting each modality as the most helpful for understanding how the AI model works. (Experts: $n = 36$, Non-Experts: $n = 108$)

short by design for non-expert users.

For the LLM-based NLE, users generally found this explanation with a more of a natural language structure and more readable and helpful. An expert participant mentions the value of LLM explanations that *"It's more helpful when textual explanations do not just describe the timestamp-wise importance but also mention the trend, or in other words, the continuity of attribution scores"*. These form of comments highlighted the need for knowing influential time windows and not only timesteps. And this is a strength of LLM-based NLEs. LLMs can often capture and describe temporal patterns automatically, identifying these informative temporal windows through rules (as in the template-based approach) is challenging and requires experimenting with different window sizes or designing convolutional filters.

A few users also mentioned they were interested in seeing more about the overall trend of each class, we believe that this is one limitation of the current approach (and not in the scope of this research where we focus on local explainability) but definitely an interesting area of exploration for future studies.

Overall, the qualitative feedback reinforces the quantitative results: users found LLM-based explanations more helpful, clear, and useful. The feedback also highlights areas for improvement for improving clarity in LLM outputs and better structuring in template-based explanations.

5 Discussion

In this section, we reflect on the findings from both the synthetic (Spike) dataset and the real-world (UCI HAR) dataset, and discuss their broader results in the context of post-hoc textual explanation generation for time series classification. Our goal was to explore two distinct approaches: template-based and LLM-based. We aimed to generate local explanations as natural language, based on attribution scores, and then evaluate their strengths, limitations, and user perception.

5.1 LLM-based NLE Prompt Engineering

The pre-study on the Spike dataset provided a controlled setup with ground truth attribution scores, allowing us to be confident about completeness and faithfulness of the attribution scores and focus on prompt engineering and improving and evaluating faithfulness, completeness, and clarity of the LLM-based NLE.

Effect of Attribution Filtering Our results from this pre-study clearly showed that passing pre-filtered influential points rather than the full raw attribution scores, significantly improved both faithfulness and completeness. Specifically, Prompt #7, which only included the high-influence points extracted from the attribution scores, achieved perfect scores across all evaluation metrics. This suggests that LLMs can generate faithful and complete explanations, but only when the input is well-structured and doesn't require the LLM to do too many numerical reasoning. For instance, when we passed all attribution scores for a single sample with 3 features and 80 time steps (resulting in 240 values), and simply asked the LLM to identify the most influential ones, it mostly failed in some cases when the scores were uniformly low or close to each other. In such cases, if there aren't a few clearly dominant attribution values, the LLM struggles to distinguish the influential ones and instead tends to hallucinate importance. This issue was more evident in class 0 samples from the Spike dataset, where almost all attribution scores were close to zero. Even when we used Prompt #6 to explicitly define thresholds and instructed the LLM that values below a certain cutoff should be considered non-influential, it still often assigned importance to small non-zero values. In some cases, it claimed that certain features were strongly influential despite attribution values being significantly below the influence threshold. This suggests that unless we constrain the input or logic tightly enough, the LLM may override instruction and try to force an specific feature as highly influential in the generated text. These findings emphasize the importance of designing prompts that either pass only influential points explicitly or reduce the LLM's decision-making responsibility in identifying them.

General knowledge hallucination on Real-World Dataset The results from the UCI HAR dataset confirmed the same patterns observed in the Spike dataset. Without passing pre-filtered influential points, the generated NLE was unfaithful and the model was faking influence according to it's general knowledge in some cases. Filtering and only passing influential points solved this effectively.
However, after adding more context about the dataset and sensor orientation of the signals, the LLM again started to hallucinate. This time, it confused attribution values with raw sensor readings. We noticed the model misinterpreting attribution scores as actual sensor values. For example, it might state, "The classifier found that the strong upward acceleration at the

beginning of the data (Time 0) led it to conclude that the user was standing." This reasoning is incorrect. The high value at Time 0 wasn't an input reading. Instead, it was a high attribution score assigned to the acceleration feature on the x-axis. This type of explanation is misleading since attribution scores only show the relative importance of a feature and time step and not any information about the actual input signal content.
To address this, we added a direct constraint to the prompt to force the LLM not to infer patterns of human motions from attribution values. This small but specific instruction was surprisingly effective and prevented these hallucinations. This result shows that LLMs, while powerful, are prone to unsupported claims unless they are explicitly bounded in the prompt. Without these types of guardrails, there is a high risk that explanations will not be correct, especially in real-world domain problems where the model has some general background knowledge.
Overall, our HAR experiments confirm that attribution filtering is essential for NLE reliability, but not sufficient. Additional instruction must be added to prevent overgeneralization, especially in domains where the LLM has strong priors.

5.2 User Perception and Survey Findings

We evaluated the two explanation styles through a user study on the HAR dataset, involving both expert and non-expert participants. Explanations were customized based on user expertise in XAI. In our participant group, survey responses tended to favor the LLM-based NLEs. LLM-based explanations received higher average ratings across all four metrics (faithfulness, completeness, clarity, and helpfulness) from both user groups compared to template-based explanations. We observed the most significant differences in clarity and helpfulness among both user groups. This suggests that, for our participants, the natural language structure and fluent reasoning of LLM explanations may have been more engaging and easier to follow.
In the evaluation of these metrics, we also observed that the clarity of an explanation can strongly influence users' judgments about faithfulness and completeness. When an explanation lacks clarity, participants are more likely to perceive it as unfaithful or incomplete as it happened with rule-based NLEs.
Moreover, in the overall textual preference question, both expert and non-expert users favored LLM-based NLE with a huge gap from the template-based NLE. Which also supports the previous claim.
A closer look at explanation modality preferences seemed to indicate an added value of textual explanations when generated by LLMs. When shown template-based NLEs, users often relied more heavily on attribution plots to understand the model's decision. But when presented with LLM-generated explanations, a substantial number of participants shifted toward preferring the *text only* or *text + plots* modalities, especially in non-experts. This may indicate that, in our study, LLM-generated NLE offered an accessible simplification that can bridge the knowledge gap for those less familiar with attribution plots or XAI methods.
Participants in our study tended to rate the combination of textual and visual explanations as the most helpful modality to understand the classifier's reasoning. This highlights the complementary nature of the two explanation formats: while plots provide a direct view of model attention, natural language helps interpret and explain that information in an appropriate way for specific end user needs.
Qualitative comments further supported preference and helpfulness of the LLM-based NLE. Users appreciated when explanations referred explicitly to important time windows or feature

trends, and several mentioned that LLM outputs felt more natural and informative.
It is notable that participants in our user study came from relatively similar educational and cultural backgrounds, with most holding a master's degree in technical or research-related fields. This background could influence their how participants interpret and value natural language explanations. For instance, more technical users may prefer attribution plots, whereas others might value textual narratives more. Since our participant pool was small and not so diverse, preferences observed in this study may not generalize to all different user groups.

5.3 Overall Comparison of Explanation Approaches

Our findings highlight a clear trade-off between the two explanation strategies. Template-based explanations offer strong guarantees on faithfulness and completeness since they are directly constructed from attribution values with a rule-based algorithm. However, they fall short in clarity and usefulness, especially for non-expert users. Their fixed template and lack of storytelling make them harder to engage with.
LLM-based explanations, in contrast, are much more user-friendly, clear, and helpful for users understanding. When guided by well-designed rules in the prompt and properly processing attribution scores, they can achieve high levels of faithfulness and completeness, while also being more natural to read and providing richer context about the real-world problem. This allows them to better help users' understanding, particularly for those without a technical background. However, they are also more vulnerable to hallucination, overgeneralization, and misinterpretation of numerical input, especially when prompt design is not handled carefully.
In our work, the most effective strategy was combining attribution preprocessing with LLM-based explanation generation. We first applied a filtering step to extract the most influential time steps and features from the attribution scores. These filtered points were then passed to the LLM using with clear instructions. This allowed us to preserve the faithfulness and completeness of attribution-based reasoning, while also generating more clear and helpful NLEs.

5.4 Limitations and Challenges

We encounter several limitations in this work:

- **Manual prompt engineering:** One major limitation in the LLM-based approach is the lack of systematic prompt optimization. Our current methodology relied heavily on manual prompt engineering and qualitative assessment of outputs. Although this was an effective approach, it was time-consuming and not fully scalable to new datasets and other LLMs. Having a labeled dataset for time series classification NLE would enable this automation and robustness.

- **Reliance on attribution methods:** Our pipeline assumes that attribution scores provided by post-hoc methods are reasonably faithful to begin with. However, attribution methods themselves have known limitations. For real-world data, this assumption of faithfulness may not be correct and introduces a layer of uncertainty in the final NLE.

- **Risk of hallucination in high-stakes domains:** Even with careful prompt engineering and attribution filtering, LLMs may occasionally produce factually incorrect reasoning

due to their nondeterministic characteristic. In high-stakes settings such as medical diagnosis, such errors could mislead users and cause harmful outcomes. While prompt engineering can reduce these errors, additional mitigation strategies should be explored, such as automated fact-checking of generated text against attribution data, hybrid text-visual interfaces that let users verify claims. These approaches could provide an extra layer of reliability.

- **Participant representativeness:** While our study included both expert and non-expert participants, the expert group was defined as individuals with academic or research experience in XAI. This group does not fully represent real-world stakeholders such as other domain specialists. Their preferences and expectations for an NLE may differ based on their specific domain. Future work can investigate more on this collaborating with domain experts and using a domain specific dataset.

- **Insufficient sensitivity to attribution sign:** In LLM-based explanations, we observed limited reasoning about the difference between negative and positive attribution values and their possible meanings, despite the importance and informativeness of these contrasting values. This reduced the helpfulness of the NLE and should be addressed through the prompt instructions.

In summary, while both explanation approaches have strengths and trade-offs, LLM-based explanations with proper prompt engineering through attribution filtering are capable of generating high-quality, user-friendly outputs. Their helpfulness advantage, especially for non-expert users, makes them a promising method for time series classification explainability to end users. However, their reliability depends on careful design decisions throughout the pipeline, and they remain vulnerable to hallucinations when used without care.

6 Conclusion

In this work, we proposed a post-hoc explanation pipeline for deep time series classification models that generates local natural language explanations (NLEs) based on attribution methods. Our results show that combining attribution-based influence scoring with a structured explanation generation step can effectively translate the model's reasoning into interpretable text.

RQ1: How to build a pipeline that can explain logical reasoning behind deep time series classification models in form of natural language?

We designed a modular four-stage pipeline: (1) classification using basic RNN and transformer models, (2) post-hoc attribution generation, (3) influence categorization through clustering, and (4) explanation generation using either a rule-based template or a prompt-guided LLM. This setup transforms attribution scores into local natural language explanations that describe the model's internal reasoning by mentioning influential parts of the time series and explain the influence that each part had on the model's decision.

RQ2: Can LLMs be guided to produce accurate and helpful NLEs based on attribution values from deep time series classifiers?

Yes. Our results show that LLMs can generate faithful explanations, but only when the attributions passed in the prompt are filtered to include only the most influential points, along with their importance scores, timestamps, and corresponding features, and when the prompts are carefully designed to guide the model's reasoning. Raw attribution values often led to hallucinations or vague reasoning, especially in cases where many scores were close to each other (noisy) or when most values were near zero. Therefore, LLM-based NLEs should be used with caution and require iterative prompt engineering and evaluation. However, we show that it is possible to design prompts that result in mostly faithful and complete NLEs.

RQ3: What are the relative strengths and weaknesses of rule-based versus LLM-generated explanations for time series classification?

Our results highlight a clear trade-off between template-based and LLM-generated NLEs. Template-based explanations are always faithful and complete (if clarity limit allows) due to their deterministic structure but often fall short in clarity and helpfulness. This was especially apparent in the user study where template-based NLEs received lower average ratings across these two metrics. Their structured format makes them reliable but less engaging, and in some cases, less helpful specially for non-expert users.

In contrast, LLM-based explanations were consistently rated higher by both expert and non-expert participants in clarity and helpfulness. Their natural language flow and ability to describe temporal trends made them more intuitive and easier to follow. These advantages were also reflected in textual explanation preferences, where nearly 78% of responses favored LLM-based NLEs, and also in the shift in modality results toward text-based or combined explanation modalities especially among non-experts. This is consistent with the modality shift patterns, where many users who had preferred plots alone with the template-based NLE switched to combined text-and-plot formats with the LLM-based NLE and where text-only explanations also became more favored suggesting greater confidence in textual explanation over visual-only. However, as mentioned earlier LLM-based NLEs are prone to hallucination or vague reasoning, particularly when attribution scores are noisy or uniformly low. So, their performance depends heavily on careful prompt design and preprocessing. Still, when properly guided, they offer strong performance in both explanation quality and user satisfaction.

RQ4: How do expert and non-expert users perceive and evaluate different forms of explana-

tions, and what explanation styles and modalities are most helpful for each group?
Based on the small sample size of our user groups, both expert and non-expert participants tended to prefer LLM-based explanations over template-based ones. Non-experts especially benefited from the natural language structure, which may helped them better understand the logic behind attribution values and connect that with their understanding of the classification model decision procedure. Across both groups, combining text and plots was considered the most helpful modality to present the explanation.
Future work could extend this pipeline to support global explanations, especially since some non-expert users stated that they are more interested in understanding overall trends in each class. Another direction is to explore how interactive explanation interfaces might better support end users in real-world applications, since users' needs and preferences are subjective and even users within the same expert or non-expert group can have different background knowledge, levels of understanding, and personal expectations. An additional interesting line of work could be to investigate how different LLMs perform on the NLE generation task compared to one another, or more interestingly, how reasoning LLMs might perform analyzing attribution values and whether they can generate faithful and complete NLEs without a need of attribution preprocessing. This methodology could also be applied to other time series machine learning tasks beyond classification, such as forecasting, anomaly detection, or segmentation.

References

[1] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[2] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[3] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, "Xai—explainable artificial intelligence," *Science robotics*, vol. 4, no. 37, p. eaay7120, 2019.

[4] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti, "Explainable ai for time series classification: a review, taxonomy and research directions," *Ieee Access*, vol. 10, pp. 100700–100724, 2022.

[5] N. Feldhus, L. Hennig, M. D. Nasert, C. Ebert, R. Schwarzenberg, and S. Möller, "Saliency map verbalization: Comparing feature importance representations from model-free and instruction-based methods," *arXiv preprint arXiv:2210.07222*, 2022.

[6] F. Sammani, T. Mukherjee, and N. Deligiannis, "Nlx-gpt: A model for natural language explanations in vision and vision-language tasks," in *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8322–8332, 2022.

[7] M. Dani, I. Rio-Torto, S. Alaniz, and Z. Akata, "Devil: Decoding vision features into language," in *DAGM German Conference on Pattern Recognition*, pp. 363–377, Springer, 2023.

[8] M. Dhaini, K. Z. Hussain, E. Zaradoukas, and G. Kasneci, "Evalxnlp: A framework for benchmarking post-hoc explainability methods on nlp models," *arXiv preprint arXiv:2505.01238*, 2025.

[9] M. Munir, S. A. Siddiqui, F. Küsters, D. Mercier, A. Dengel, and S. Ahmed, "Tsxplain: demystification of dnn decisions for time-series using natural language and statistical features," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*, pp. 426–439, Springer, 2019.

[10] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: a review and experimental evaluation of recent time series classification algorithms," *Data Mining and Knowledge Discovery*, vol. 38, no. 4, pp. 1958–2031, 2024.

[11] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[12] J. Faouzi, "Time series classification: A review of algorithms and implementations," *Machine Learning (Emerging Trends and Applications)*, 2022.

[13] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 637–646, 2017.

[14] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585, IEEE, 2017.

[15] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[16] A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier, "Deep learning for time series classification using new hand-crafted convolution filters," in *2022 IEEE International Conference on Big Data (Big Data)*, pp. 972–981, IEEE, 2022.

[17] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "Lite: Light inception with boosting techniques for time series classification," in *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, IEEE, 2023.

[18] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[23] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2114–2124, 2021.

[24] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, "Gated transformer networks for multivariate time series classification," *arXiv preprint arXiv:2103.14438*, 2021.

[25] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[26] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, pp. 3319–3328, PMLR, 2017.

[27] U. Schlegel and D. A. Keim, "Time series model attribution visualizations as explanations," in *2021 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*, pp. 27–31, IEEE, 2021.

[28] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, pp. 107–144, 2007.

[29] J. Enguehard, "Time interpret: a unified model interpretability library for time series," *arXiv preprint arXiv:2306.02968*, 2023.

[30] S. Tonekaboni, S. Joshi, K. Campbell, D. K. Duvenaud, and A. Goldenberg, "What went wrong and when? instance-wise feature importance for time-series black-box models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 799–809, 2020.

[31] J. Crabbé and M. Van Der Schaar, "Explaining time series predictions with dynamic masks," in *International conference on machine learning*, pp. 2166–2177, PMLR, 2021.

[32] J. Enguehard, "Learning perturbations to explain time series predictions," in *International Conference on Machine Learning*, pp. 9329–9342, PMLR, 2023.

[33] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, pp. 3319–3328, PMLR, 2017.

[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833, Springer, 2014.

[35] M. Kayser, O.-M. Camburu, L. Salewski, C. Emde, V. Do, Z. Akata, and T. Lukasiewicz, "e-vil: A dataset and benchmark for natural language explanations in vision-language tasks," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1244–1254, 2021.

[36] S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, and S. Ahmed, "Tsviz: Demystification of deep learning models for time-series analysis," *IEEE Access*, vol. 7, pp. 67027–67040, 2019.

[37] J. Höllig, S. Thoma, and F. Grimm, "Xtsc-bench: quantitative benchmarking for explainers on time series classification," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, pp. 1126–1131, IEEE, 2023.

[38] C. Leiter, P. Lertvittayakumjorn, M. Fomicheva, W. Zhao, Y. Gao, and S. Eger, "Towards explainable evaluation metrics for natural language generation," *arXiv preprint arXiv:2203.11131*, 2022.

[39] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra, "Human Activity Recognition Using Smartphones." UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C54S4K.

[40] D. Anguita, A. Ghio, L. Oneto, F. X. Llanas Parra, and J. L. Reyes Ortiz, "Energy efficient smartphone-based activity recognition using fixed-point arithmetic," *Journal of universal computer science*, vol. 19, no. 9, pp. 1295–1314, 2013.

A LLM Prompts Used in NLE Generation

A.1 Spike Dataset Prompts

To highlight how prompt design evolved, we report the numbered prompt versions used on a class 1 sample from the Spike dataset. In each version, the bolded part shows the part that was modified.

Prompt #1 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The attribution scores are as follows: [['-0.0', '-0.01', ..., '0.0'], ['0.0', '-0.0', ..., '0.0'], ['-0.0', '0.0', ..., '0.0']]
The classifier predicted the sample to belong to class 1. Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #2 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The attribution scores are as follows:
For feature 0: {**0: -0.0, 1: -0.01, 2: -0.01, ..., 43: 11.84, ... 79: 0.0**}
For feature 1: {**0: 0.0, ..., 43: -2.59, ..., 79: 0.0**}
For feature 2: {**0: -0.0, ..., 43: 0.04, ..., 79: 0.0**}
Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #3 A deep time series classifier has been trained on labeled data. After training, a **Temporal Integrated Gradients** method was applied to compute attribution values for each time step of each feature of a specific input sample. **The maximum possible attribution is +inf, and the minimum is -inf.**
The attribution scores are as follows:
For feature 0: {0: -0.0, 1: -0.01, 2: -0.01, ..., 43: 11.84, ... 79: 0.0}
For feature 1: {0: 0.0, ..., 43: -2.59, ..., 79: 0.0}
For feature 2: {0: -0.0, ..., 43: 0.04, ..., 79: 0.0}
Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #4 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The attribution scores are as follows:
For feature 0: {0: -0.0, 1: -0.01, 2: -0.01, ..., 43: 11.84, ... 79: 0.0}
For feature 1: {0: 0.0, ..., 43: -2.59, ..., 79: 0.0}
For feature 2: {0: -0.0, ..., 43: 0.04, ..., 79: 0.0}
Only if an attribution value is higher than 1.126, it is considered influential. The classifier predicted the sample to belong to class 1.

Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise. **List at most the top 3 most influential feature-time step combinations.**

Prompt #5 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
To aid interpretation, attribution values have been grouped into three categories based on their absolute magnitude:
No influence: $0.0 \leq |\textbf{value}| < 1.12$**,**
Slightly influential: $1.13 \leq |\textbf{value}| < 6.46$**,**
Strongly influential: $6.46 \leq |\textbf{value}| < 15.30$
The attribution scores are as follows:
For feature 0: {0: -0.0, 1: -0.01, 2: -0.01, ..., 43: 11.84, ... 79: 0.0}
For feature 1: {0: 0.0, ..., 43: -2.59, ..., 79: 0.0}
For feature 2: {0: -0.0, ..., 43: 0.04, ..., 79: 0.0}
The classifier predicted the sample to belong to class 1. Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #6 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The attribution scores are as follows:
For feature 0: {0: -0.0, 1: -0.01, 2: -0.01, ..., 43: 11.84, ... 79: 0.0}
For feature 1: {0: 0.0, ..., 43: -2.59, ..., 79: 0.0}
For feature 2: {0: -0.0, ..., 43: 0.04, ..., 79: 0.0}
Only if an attribution value is higher than 1.13, it is considered influential. The classifier predicted the sample to belong to class 1.
Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #7 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample. The classifier predicted the sample to belong to class 1.
The following features and time steps were identified as influential:
- Feature 0, Time 43 (Strongly Influential, Attribution = 11.84)
- Feature 1, Time 43 (Moderately Influential, Attribution = 2.59)
Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

A.2 UCI HAR Dataset Prompts

we report the numbered prompt versions used on a class STANDING sample from the UCI HAR dataset. In each version, the bolded part shows the part that was modified.

Prompt #1 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The attribution scores are as follows (formatted as time_step: attribution_value for each feature):
For body_acc_x: 0:0.18, 1:-0.04, 2:-0.04, ..., 127:0.1
For body_acc_y: 0:0.9, 1:0.82, ..., 124:-0.02
For body_acc_z: 0:0.14, 1:0.17, ..., 127:-0.03
...
For total_acc_z: 0:-0.25, 1:0.03, ..., 121:0.03
The classifier predicted the sample to belong to class STANDING. Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #2 A deep time series classifier has been trained on labeled data. After training, an attribution-based explanation method was applied to compute attribution values for each time step of each feature of a specific input sample.
The following features and time steps were identified as influential:
- Feature: total_acc_x
- Time 0: Strongly Influential (Attribution value: 9.28)
- Feature: total_acc_y
- Time 0: Moderately Influential (Attribution value: 2.77)
The classifier predicted the sample to belong to class STANDING. Explain the classifier's prediction for this input. Focus on the most influential features and time steps based on the attribution scores. Keep the explanation concise.

Prompt #3 and #4 A deep time series classifier has been trained on the UCI Human Activity Recognition dataset. **This dataset includes raw inertial signals collected from a Samsung Galaxy S2 smartphone's accelerometer and gyroscope sensors.**
Each input sample includes 9 signals (body acceleration, gyroscope, and total acceleration along x, y, z axes) over 128 time steps.After training, a temporal integrated gradients method was applied to compute attribution values for each time step of each feature of a specific input sample.
A phone case and belt was used to mount the smartphone to the user's waist. The smartphone is in portrait mode with the screen facing outward and the top pointing up. In this position, sensor orientation in the dataset is as follows:
- x-axis: vertical, pointing upward (bottom to top of phone)
- y-axis: horizontal, pointing left (across the screen)
- z-axis: outward, perpendicular to the screen (toward the viewer)
The following features and time steps were identified as influential:
- Feature: 'total_acc_x' - Time 0: Strongly Influential (Attribution value: 9.28)
- Feature: 'total_acc_y' - Time 0: Moderately Influential (Attribution value: 2.77)

The classifier predicted the sample to belong to class STANDING. Explain the classifier's prediction for this sample using the attribution scores and mentioning influential features and time steps. Keep the explanation concise.

Prompt #4 adds the following sentence: Do NOT infer sensor magnitudes or human motions. Your explanation must mention importance, not what happened in the signal.

B User Survey Instructions

Participant Instruction Handout

Task Overview

We want to understand which type of explanation helps you [Expert only: (as an expert user)][Non-expert only: (as a non-expert user)]better understand how an AI model makes its decisions.

This explanation aims to give you [Expert only: a good and detailed] [Non-expert only: an overall] understanding of how the AI model is working on a given sample.

The AI model in this study is a classification model that receives motion sensor data as input and classifies an activity. It analyzes short sequences of motion data collected from a smartphone's accelerometer and gyroscope while a person performs daily activities (e.g., walking, sitting, standing). The phone was worn on the waist, so each signal feature (e.g., body acc x) measures movement in one direction relative to the person's body.

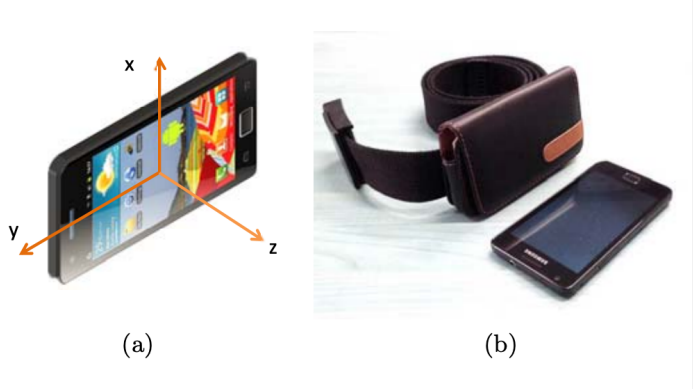

Figure 1: Smartphone placement on the waist and sensor axis orientation [40]

Note: You can refer back to Figure 1 at any time during the survey to understand the sensor axis orientations of the input data.

What You'll See

You will review 12 samples, one per page. For each sample, you will see plots of the raw signals and two short text explanations of the model's decision.

Each page shows:

- **Sample Image:** On top of the page you'll see an image with two plots:
  - Top plot: A time-series plot of the sensor readings (the input sample to the AI model).
  - Bottom plot: A plot showing [Expert only: attribution values extracted from an attribution-based method. We categorized attribution values into three influence levels and][Non-expert only: how much each feature and time step

influenced the AI model's prediction.]Only moderately and highly influential points are shown. Feature influence levels are visualized with shaded bands on this plot — orange for slightly influential and red for strongly influential values.

- **Two Text Explanations:** Brief descriptions of why the model made its prediction and what were the most influential parts of the input.
- **Ratings & Preferences:** You'll rate each explanation on four criteria and indicate which format you prefer.

Evaluation Metrics

You will rate each textual explanation on the following dimensions (using a 1–5 scale):

- **Faithfulness:** Does the explanation point to the right parts of the plots—highlighting the influential time steps and features correctly? You can check this by looking at the bottom plot.
- **Helpfulness:** Did this textual explanation help you understand why the model made its choice?
- **Completeness:** Does the textual explanation cover all the most important parts that influenced the decision?
- **Clarity:** Is the explanation easy to read and understand for you?

Your Task

1. Examine the plots and read both text explanations.
2. Rate each text explanation on the four metrics (1–5 scale).
3. Choose the form of explanation you found most useful and preferred in that sample.

Thank you so much for your participation!

C Survey User Interface

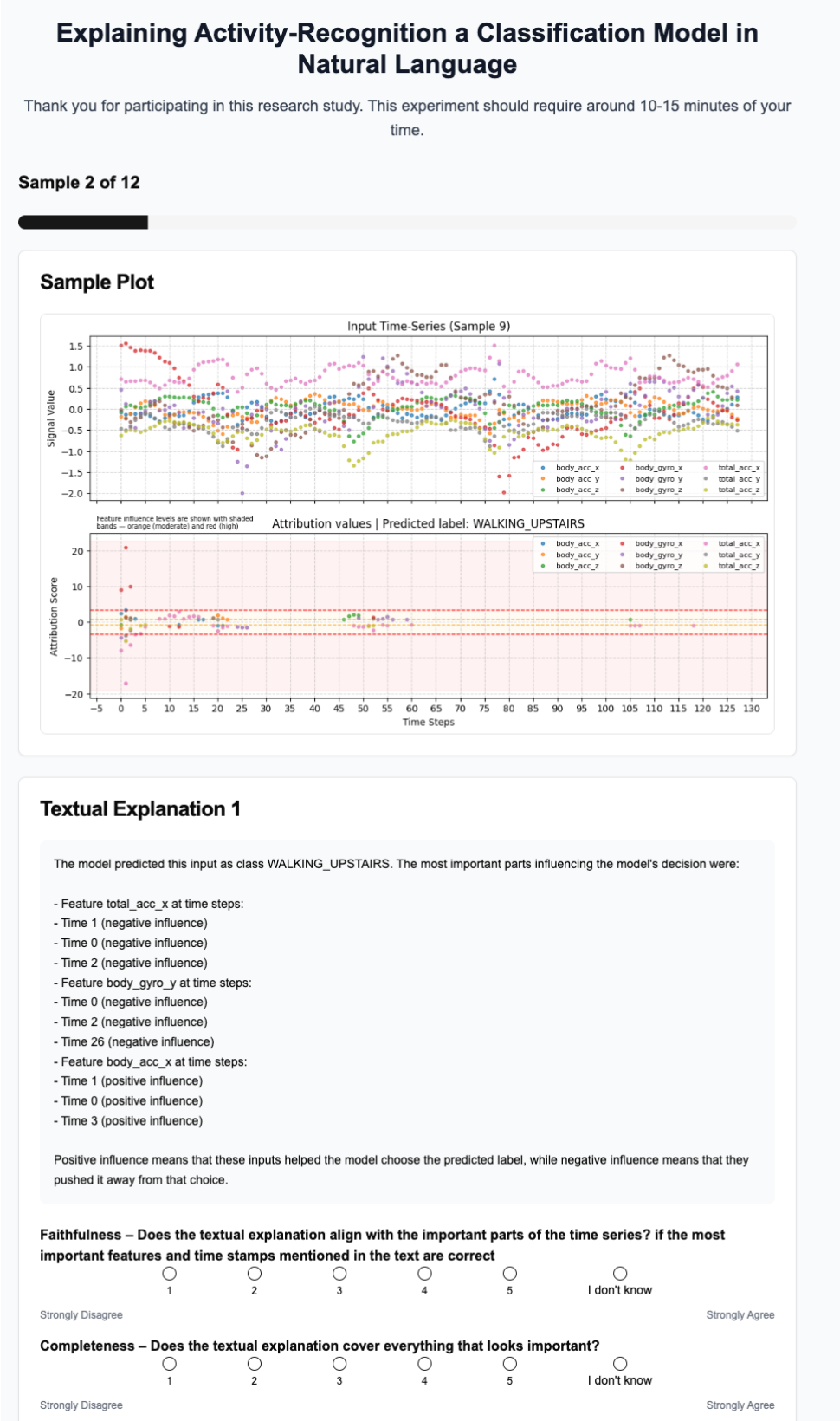

Figure 9: Survey Page (Top Half)

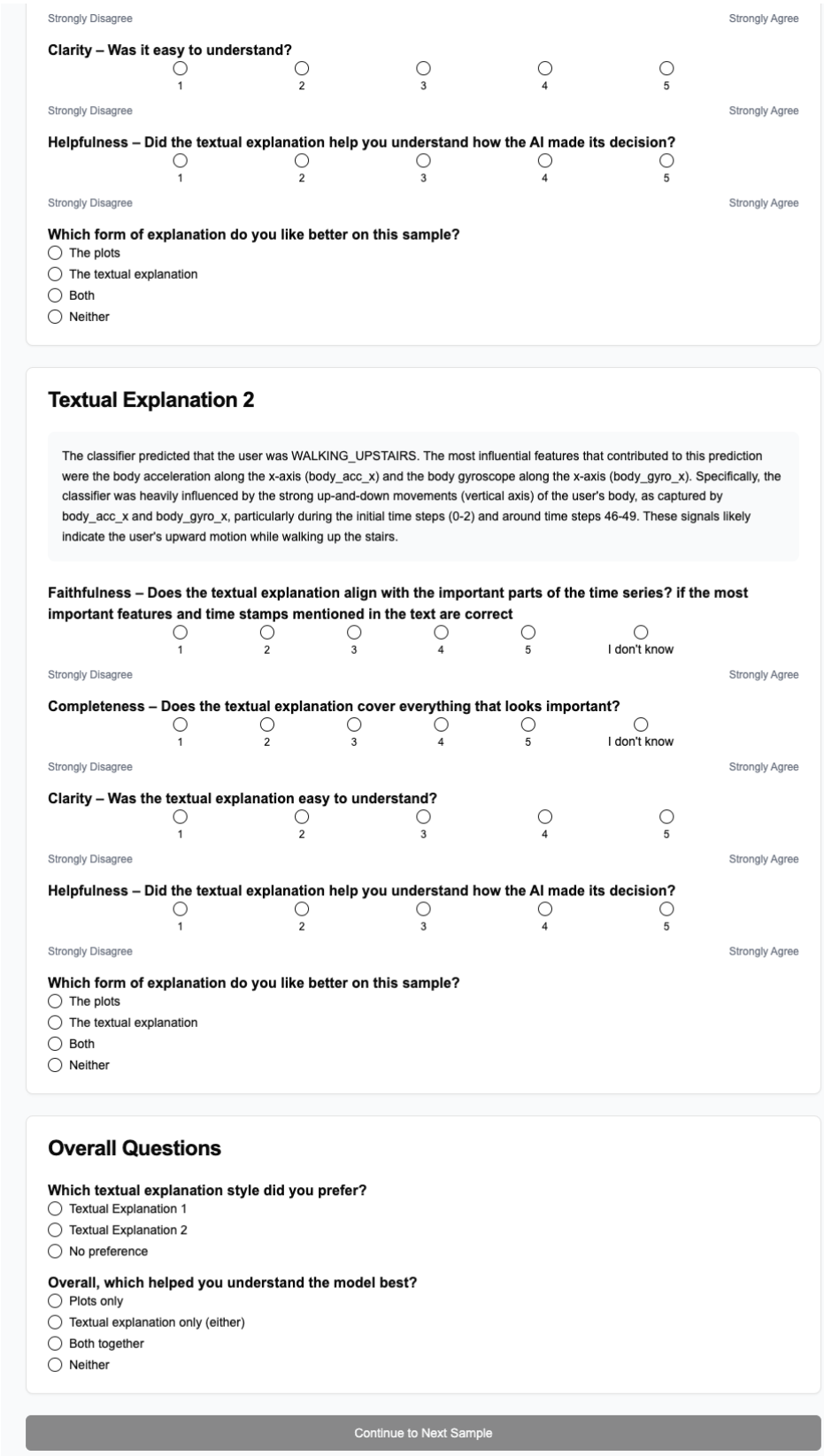

Figure 10: Survey Page (Bottom Half)