TIMESAE: SPARSE DECODING FOR FAITHFUL EXPLANATIONS OF BLACK-BOX TIME SERIES MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

As black box models and pretrained models gain traction in time series applications, understanding and explaining their predictions becomes increasingly vital, especially in high-stakes domains where interpretability and trust are essential. However, most of the existing methods involve only in-distribution explanation, and do not generalize outside the training support, which requires the learning capability of generalization. In this work, we aim to provide a framework to explain black-box models for time series data through the dual lenses of Sparse Autoencoders (SAEs) and causality. We show that many current explanation methods are sensitive to distributional shifts, limiting their effectiveness in real-world scenarios. Building on the concept of Sparse Autoencoder, we introduce `TimeSAE` , a framework for black-box model explanation. We conduct extensive evaluations of `TimeSAE` on both synthetic and real-world time series datasets, comparing it to leading baselines. The results, supported by both quantitative metrics and qualitative insights, show that `TimeSAE` delivers more faithful and robust explanations. Our code is available in an easy-to-use library `TimeSAE` -Lib: https://anonymous.4open.science/w/TimeSAE-571D/.

1 INTRODUCTION

The rise of black box models such as large foundation models has revolutionized various fields, including time series analysis, with applications in finance (Bento et al., 2021), healthcare (Kaushik et al., 2020), and environmental science (Adebayo et al., 2021). These networks often make critical decisions, especially in sensitive domains where decisions based on forecasting outcomes such as managing grid stability in Energy (Eid et al., 2016), Healthcare (Dairi et al., 2021), yet the underlying decision-making process is difficult to interpret due to the black-box nature of the models. This opacity has motivated the rise of explainable AI (XAI) techniques to provide human-understandable explanations for model decisions. While XAI has been predominantly applied in image classification, it is extending into other fields, such as audio and time series (Parekh et al., 2022; Queen et al., 2023).

Current methods in enhancing explainability for time series primarily identify key signal locations (sub-instance) affecting model predictions. For instance, Shi et al. (2023) uses LIME (Ribeiro et al., 2016) to explain water level prediction models. Additionally, perturbation methods like Dynamask (Crabbé & Van Der Schaar, 2021) and Extrmask (Enguehard, 2023) modify less critical features to evaluate their impact but often struggle with feature interdependencies and generalization. Despite their insights, these techniques face challenges with Out-of-Distribution (OOD) samples, affecting the *faithfulness* of explanations (Queen et al., 2023).

Explaning time series black-box models requires the ability to generalize beyond the training distribution, which is essential for the robust deployment of explanatory algorithms in real-world scenarios. In addressing the extrapolation of explanation, Queen et al. (2023) retrain a white-box model for consistency, though this depends on knowing the model's structure and may not ensure consistent explanations. Similarly, Liu et al. (2024b) uses a stochastic mask to tackle OOD issues; however, challenges remain as explanations are still treated as OOD and lack clarity, with explanation compositionality aspects unaddressed. Furthermore, *faithfulness* is also a desirable property of any explanation method, broadly defined as the ability of the method to provide accurate descriptions of the underlying reasoning (Gat et al., 2023; Jain & Wallace, 2019). Formally, the *faithfulness* challenge can be described as follows:

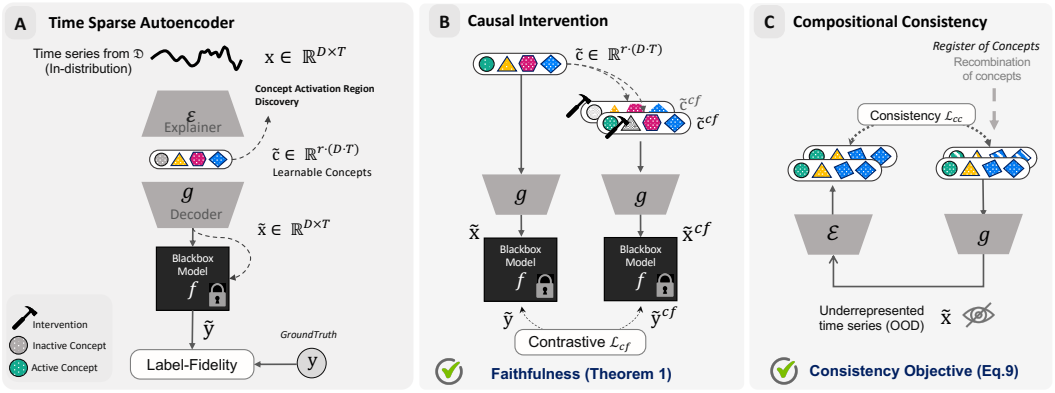

Figure 1: **Overview of Time Series Sparse Autoencoder (`TimeSAE`):** **(A)** The framework assumes access to a black-box model f and aims to explain its predictions on data $\mathbf{x} \in \mathcal{X}$ by learning an explainer $\mathcal{E}$ and a decoder $\boldsymbol{g}$ that decompose the time series into interpretable components. **(B)** For faithfulness, the sparse autoencoder $(\mathcal{E}, \boldsymbol{g})$ incorporates properties to leverage counterfactual explanations. A contrastive learning loss incorporates a set of counterfactuals $\tilde{\mathbf{x}}^{cf}$, obtained by intervening on concepts and which produce, via f, a contradictory label $\tilde{\mathbf{y}}^{cf}$ relative to the original $\tilde{\mathbf{y}}$. **(C)** To ensure compositional explanations, the method enforces the explainer $\mathcal{E}$ to generate consistent explanations by combining intermediate findings.

Definition 1 (**Faithful Explanations.**). *Let $f : \mathcal{X} \to \mathcal{Y}$ be a black-box model, consider an input $\mathbf{x} \in \mathcal{X}$, and let $\mathcal{E} : \mathcal{X} \to \mathcal{C}$ be an explainer that generates explanations in the concept space $\mathcal{C}$.* Faithfulness *is the ability of $\mathcal{E}$ to accurately reflect f's reasoning, i.e the predictive capability of $f(\mathbf{x})$ from the explanations $\mathcal{E}(\mathbf{x})$.*

To address Definition 1, there is often confusion between two types of model explanations: **a)** *what knowledge does the model encode?* **b)** *Why does it make certain predictions?* Here, the *"what"* aspect does not directly explain the model's decision-making, as not all encoded features are necessarily used in predictions. Explanations based on the *"what"* are typically correlational, rather than causal. Understanding a model's reasoning requires attention to the *"why"*, making causality indispensable, which is defined as how changes in inputs impact the outputs. Indeed, counterfactuals (CFs), which explore *"what if"* scenarios by identifying minimal and feasible changes to inputs that alter predictions, are at the highest level of Pearl's causal hierarchy (Pearl, 2009), highlighting how changes lead to a different prediction. To truly understand a model's reasoning, it is essential to focus on the *"why"*, making causality a key component of *faithfulness*.

Contributions We propose `TimeSAE` , a Sparse Autoencoder (SAE) for learning explanation for time series black-box models. Specifically, we replace the need of the sub-instances via masking time steps and features by directly learning an end-to-end SAE using JumpReLU (Rajamanoharan et al., 2024b) to explain the prediction by concepts, using explaination-embedded instances that are close to the original distribution while maintaining label-prediction. Our method is model-agnostic and operates in a post-hoc manner, requiring no access to the internal structure, parameters, or intermediate activations of the model to explain. We summarize our main contributions as:

[1] We examine the shortcomings of current explanation techniques for time series models through the lens of Concept Learning and Causal Counterfactual. To address these, we introduce `TimeSAE` , a framework leveraging a SAE for Concept Learning of Time Series, and mitigates distribution shift by generating samples from learned concepts aligning with the original distribution.

[2] To ensure a *faithful explanation* of `TimeSAE` 's outputs, we provide a theoretical guarantee that its structure can approximate counterfactual explanations. In particular, it can identify which dictionary atoms would lead the black-box model to produce an alternative prediction.

[3] We evaluate our method on eight time series datasets, demonstrating its superior performance to existing explainers. We further highlight its effectiveness in a real-world case, and made the code available in an easy-to-use library `TimeSAE` -Lib[1].

[1] https://anonymous.4open.science/r/TimeSAE-571D/

[4] We introduce *EliteLJ*, a new open-source dataset designed to benchmark time-series explanation methods. The dataset consists of skeleton-based motion data from long jump athletes and includes expert annotations labeling key phases (e.g., run-up, take-off, flight, landing) and qualitative assessments (e.g., good vs. bad take-off, correct vs. incorrect landing posture). This combination provides a realistic use case for evaluating explainability methods on sports performance data.

2 Related Work

Concepts-based XAI and Sparse Autoencoders. Concept-based Interpretable Networks (CoINs) encompasses models using human-interpretable concepts for prediction (Parekh et al., 2025) for white box models. Existing work in vision specifies concepts using either human supervision to select and provide their concept labels (Koh et al., 2020) or extracting them automatically with large language models (Oikarinen et al., 2023). Other works exploit unsupervised methods to automatically discover concepts (Alvarez Melis & Jaakkola, 2018; Sarkar et al., 2022) in by-design approaches and some of them (Parekh et al., 2021) leverage sparsity and diversity constraints directly on the concept activations, which is close to the approach adopted in this paper. Unlike existing unsupervised concept learning methods, which only emphasize properties such as *faithfulness*, we also focus on *causality* and *compositionality* of concepts, i.e. concepts being presents simultaneously and composed to form complex patterns, including in time series. Following Mikolov et al. (2013) on compositional word vectors, researchers have increasingly investigated how deep learning models exhibit compositional behavior (Brady et al., 2023; Wiedemer et al., 2024).

Counterfactual Explanations. Counterfactual explanations can be generated with various guidances (Ye & Keogh, 2009; Bahri et al., 2024; Li et al., 2024; Tonekaboni et al., 2020; Li et al., 2023; Wang et al., 2023). However, distributional shift remains a critical issue. To address this, Liu et al. (2024c) generate in-domain perturbations with contrastive learning. Liu et al. (2024b) propose an information bottleneck-based objective function to ensure model faithfulness while avoiding distributional shifts. We address this by generating samples from learned concepts aligning with the original data distribution.

Temporal Interactions in Explanations. Due to the specific nature of black-box model time series explanation, in both classification and regression, effects of time step must be taken into account. Leung et al. (2023) aggregate the impact across subsequent time windows while Tonekaboni et al. (2020) and Suresh et al. (2017) quantify the significance of a time step by measuring its effect on model prediction. Despite these advances, challenges remain in fully addressing the complex interactions between observational points. Yang et al. (2024) introduce time-step interactions, and Märtens & Yau (2020) propose "Variance decomposition" to quantify the variance attributed to each component. In our work, we propose a decomposition of the decoder that models and links these interactions to concepts for enhanced explanations and interpretability.

3 A Time Series Sparse Autoencoder for Faithful Explanations

To achieve an accurate explanation, we propose `TimeSAE` , a framework leveraging a Sparse Autoencoder that maintains the informativeness of time series for the black-box model. We show how $\mathcal{E}$ inverts the decoder $\boldsymbol{g}$ for more robust concept explanations, while generating approximate CFs for more faithful explanations. At the sequence level, `TimeSAE` offers a feature-level decomposition using the temporal decoder $\boldsymbol{g}$, which maps concept activations to their corresponding score activations.

Notation. This work focuses on explainability in time series for both regression and classification. Throughout this work, a time series instance $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{D \times T}$ is represented by a $D \times T$ real-valued matrix, where T is the number of time steps in the series, and D is the feature dimension. If $D > 1$, the time series is multivariate; otherwise, it is univariate. We denote by $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i \in [N]\}$ the training set, which consists of N instances $\mathbf{x}_i$ along with their associated labels $\mathbf{y}_i$, where $\mathbf{y}_i \in \mathcal{Y}$ and $\mathcal{Y}$ is the set of all possible continuous or discrete labels. A black-box time series predictor $f(\cdot)$ takes an instance $\mathbf{x} \in \mathcal{X}$ as input, and outputs a label $f(\mathbf{x}) \in \mathcal{Y}$. We further define an overcomplete sparse autoencoder $(\mathcal{E}, \boldsymbol{g})$, denoted for simplicity by SAE, where $\mathcal{E} : \mathcal{X} \to \mathcal{C}$, designed to extract concept explanations, with $\mathcal{C}$ the concept space. The activated concepts $\mathbf{c} \in \mathcal{C}$ are mapped back in the input space through the decoder $\boldsymbol{g} : \mathcal{C} \to \mathbb{R}^{D \times T}$, resulting in an *explanation-embedded instance*

$\widetilde{\mathbf{x}} \in \mathbb{R}^{D \times T}$. For a positive integer n, let $[n] = \{1, \cdots, n\}$ denote the set of integers from 1 to n, and we use $|\cdot|$ to represent either the dimension (or length) of a vector, or the cardinality of a set.

3.1 Sparse Autoencoder

The Sparse Autoencoder, defined by a pair of encoder $\mathcal{E}$ and decoder $\boldsymbol{g}$ functions, decomposes each input $\mathbf{x}$ into a sparse combination of learned feature directions $\mathbf{c}$, from a dictionary $\mathbf{M}$. It then generates the reconstruction $\tilde{\mathbf{x}}$ of the input. This process is summarized as:

$$\mathbf{c} := \mathcal{E}(\mathbf{x}) = \sigma(\mathbf{M}\mathbf{x} + \mathbf{b}), \quad \tilde{\mathbf{x}} := \boldsymbol{g}\big(\mathbf{c}\big), \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{d \times (D \cdot T)}$, with $d := |\mathbf{c}| = r \cdot (D \cdot T)$, normalized row-wise to control sparsity. Here, r is a hyperparameter controlling the size of d, and $\mathbf{b} \in \mathbb{R}^d$ are learned parameters. In such expression, $\mathcal{E}(\mathbf{x})$ is a sparse, non-negative vector of feature magnitudes present in the input activation. The columns of $\mathbf{M}$ represent the learned feature directions that form the dictionary used by the SAE for decomposition. The activation function σ varies: ReLU or gated (Rajamanoharan et al., 2024a) in some cases, while TopK SAEs (Makhzani & Frey, 2013) keep only the top-k active concepts. Unfortunately, we find that this may often result in "dead" concepts similar to the phenomenon observed in LLMs (Gao et al., 2024), where some components don't actively contribute at all from the start of learning. A detailed discussion is provided in Appendix B.1.2. To further boost fidelity, Rajamanoharan et al. (2024b) propose the JumpReLU activation, denoted as JumpReLU_ϕ, which extends the standard ReLU-based SAE architecture (Ng, 2011) by incorporating an additional positive, vector-valued learnable parameter $\boldsymbol{\phi} \in \mathbb{R}+^{d \times (D \cdot T)}$. JumpReLU uses a learnable threshold ϕ_k for each concept feature k, and a feature is active only if the encoder output exceeds ϕ_k. This relaxation prevents *"dead"* activations, stabilizing concept learning and improving reconstruction fidelity.

Loss functions Within our framework `TimeSAE`, we generate the reconstructed instance $\tilde{\mathbf{x}}$ through $\text{JumpReLU}_\phi(\mathbf{c})$, which guarantees better learning of the concept dictionary and minimizes error reconstruction. Formally, our objective function can be defined as:

$$\mathcal{L}_{\text{SAE}}(\mathbf{x}; \mathcal{E}, \boldsymbol{g}) := \underbrace{||\mathbf{x} - \boldsymbol{g}\big(\mathcal{E}(\mathbf{x})\big)||_2^2}_{\mathcal{L}_{\text{rec-fidelity}}} + \underbrace{\eta\, \mathbf{s}\big(\mathcal{E}(\mathbf{x})\big)}_{\mathcal{L}_{\text{sparsity}}}, \tag{2}$$

where $\mathbf{s}(\cdot)$ is a function of the concepts' activations that penalises the non-sparse decompositions (i.e., $\text{JumpReLU}_\phi(\mathbf{c})$) and the *sparsity coefficient* η controls the trade-off between sparsity and reconstruction fidelity.

Temporal Concept Learning In `TimeSAE`, the encoder and decoder use *Block Temporal Convolutional Networks (TCN)* (Bai et al., 2018) to capture temporal dependencies. The encoder maps inputs to a sparse latent space, normalized for stable training, and applies JumpReLU_ϕ with learnable thresholds for robust concept activations. *Squeeze-and-Excitation (SE)* blocks (Hu et al., 2018) adaptively reweight feature channels. The decoder mirrors this design, reconstructing outputs through TCN layers with normalization, SE, and ReLU. Full architectural details are in Appendix B.5.

3.2 Faithfulness via Counterfactual Explanations

To measure the causal effect of a concept on the model prediction, we rely on the *Causal Concept Effect (CaCE)* (Goyal et al., 2019) which we formally describe as follows.

Definition 2 (CaCE (Goyal et al., 2019)). *Given an intervention $I_k : \boldsymbol{c}_k \mapsto \boldsymbol{c}'_k$, a black-box model $f : \mathcal{X} \to \mathcal{Y}$ and a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i \in [N]\}$ of size N, the Causal Concept Effect (CaCE) is:*

$$\text{CaCE}_f(I_k) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}\left[f(\mathbf{x}) | do(\boldsymbol{c}_k = I_k(\boldsymbol{c}_k))\right] - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}\left[f(\mathbf{x}) | do(\boldsymbol{c}_k = \boldsymbol{c}_k)\right]. \tag{3}$$

The causal effect and explanation of a model are both related to counterfactuals (CFs). This enables causal estimation in a model-agnostic manner as CFs can be obtained using only the explainer $\mathcal{E}$. We can now define the Approximated Counterfactual Explanation:

Definition 3 (Approximated Counterfactual Explanation (Gat et al., 2023)). *Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i \in [N]\}$ of size N, an explainer $\mathcal{E} : \mathcal{X} \to \mathcal{C}$ and an intervention $I_k : \boldsymbol{c}_k \mapsto \boldsymbol{c}'_k$, the approximated counterfactual explanation $\mathcal{E}^{cf}$ is defined to be:*

$$\mathcal{E}^{cf} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \mathcal{E}(\tilde{\mathbf{x}}_{\boldsymbol{c}'_k}) - \mathcal{E}(\tilde{\mathbf{x}}_{\boldsymbol{c}_k}), \tag{4}$$

where $\tilde{\mathbf{x}}_{\boldsymbol{c}'_k}$ is the explanation-embedded instance after intervention, and $\tilde{\mathbf{x}}_{\boldsymbol{c}_k}$ before intervention.

In Appendix A, we provide a discussion of the Approximated CF Explanation methods. Faithfulness, as outlined in Definition 1, remains only partially addressed by the proposed explainable model. While it promotes order-preserving diversity, it does not directly address how faithful it is. In Gat et al. (2023), counterfactuals with causal relationships are shown to ensure faithfulness of explanation by validating that higher-ranked interventions have greater causal effects. Building on this, we prove the following result, showing that our explanation method preserves the relative ordering of causal effects under bounded approximation error, defined as *order-faithfulness*.

Theorem 1 (**Faithfulness in Sparse Autoencoder-Based Approximate Counterfactuals).** *Let* $\mathbf{x}$ *be a time-series input and* f *a black-box model whose true output is* $\mathbf{y} = f(\mathbf{x})$. *Suppose* $(\mathcal{E}, \boldsymbol{g})$ *is an encoder-decoder, where* $\mathcal{E}$ *encodes* $\mathbf{x}$ *to latent concepts, and* $\boldsymbol{g}$ *decodes these concepts into* $\widetilde{\mathbf{x}} = \boldsymbol{g}(\mathcal{E}(\mathbf{x}))$ *such that* $\forall \mathbf{x} \in \mathcal{D}$, $f(\widetilde{\mathbf{x}}) \approx f(\mathbf{x})$. *For an intervention, define an* approximate counterfactual *by altering concepts* $\mathbf{c} \mapsto \mathbf{c}^{cf}$, *and let* $\widetilde{\mathbf{x}}^{cf} = \boldsymbol{g}\big(\mathbf{c}^{cf}\big)$. *Assume that*

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[\,|\, f(\widetilde{\mathbf{x}}^{cf}) - \mathbf{y}^{cf}\,|\,\big] \;\leq\; \epsilon_{cf}, \tag{5}$$

where $\mathbf{y}^{cf}$ *is the "true" counterfactual label (i.e., what* $f(\mathbf{x})$ *would be under the exact causal intervention), and* ε_{cf} *is a small approximation error. Then, for any pair of interventions* $I_1 : c_1 \mapsto c_1'$ *and* $I_2 : c_2 \mapsto c_2'$, *if the* true *causal effects satisfy*

$$\texttt{CaCE}_f(I_1, c_1, c_1') \;>\; \texttt{CaCE}_f(I_2, c_2, c_2'), \tag{6}$$

there exists a sufficiently small ϵ_{cf} *so that*

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f(\widetilde{\mathbf{x}}^{cf}_{I_1}) - f(\widetilde{\mathbf{x}})\Big] \;>\; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f(\widetilde{\mathbf{x}}^{cf}_{I_2}) - f(\widetilde{\mathbf{x}})\Big], \tag{7}$$

where $\widetilde{\mathbf{x}}^{cf}_{I_1}$ *and* $\widetilde{\mathbf{x}}^{cf}_{I_2}$ *are the explanation-embedded instances respectively obtained after interventions* I_1 *and* I_2. *This preserves the* ordering *of causal effects, i.e. order faithfulness.*

Proof Sketch. We define the actual and approximate causal effects of interventions and limit approximation errors. By ensuring that the combined approximation and reconstruction errors are smaller than the actual causal effects, we guarantee that order is preserved. Thus, approximate counterfactuals based on a sparse autoencoder preserve the causal effects' order. The full proof is provided in Section A.

Practical Implementation. To enforce faithfulness, we use a contrastive loss InfoNCE (Oord et al., 2018) during the training of the Sparse Autoencoder. We define the positive pairs as counterfactuals from the same intervention I_k, while negative pairs come from different interventions I_j, ensuring $\texttt{CaCE}_f(I_k) > \texttt{CaCE}_f(I_j)$. The contrastive loss is formulated as:

$$\mathcal{L}_{cf} = -\log\left(\frac{\exp(\text{sim}(f(\widetilde{\mathbf{x}}^{cf}_{I_k}), f(\widetilde{\mathbf{x}'}^{cf}_{I_k}))/\tau)}{\sum_j \exp(\text{sim}(f(\widetilde{\mathbf{x}}^{cf}_{I_k}), f(\widetilde{\mathbf{x}}^{cf}_{I_j}))/\tau)}\right), \tag{8}$$

where sim$(\cdot,\cdot)$ is the cosine similarity and τ is a temperature parameter that controls the sharpness of the similarity distribution.

3.3 Generalization in Out-of-Distribution

For a generator $\boldsymbol{g}$ that is consistent for an OOD sample, Wiedemer et al. (2024) showed that an autoencoder will *slot-identify* concepts $\mathbf{c}$ in $\mathcal{C}$. The conditions on the encoder discussed in the previous section aim to ensure that $\mathcal{E}$ inverts $\boldsymbol{g}$ over the entire input space $\mathcal{X}$. This behavior is encouraged within the training space $\mathcal{X}$ (in-distribution) by minimizing the *reconstruction fidelity* objective $\mathcal{L}_{\text{rec-fidelity}}$. However, no mechanism is in place to enforce this inversion behavior of $\mathcal{E}$ on $\boldsymbol{g}$ also outside of $\mathcal{X}$ (out-of-distribution). To address this, we propose to use the following compositional consistency loss (Wiedemer et al., 2024):

$$\mathcal{L}_{cc}(\mathcal{E}, \boldsymbol{g}, \mathcal{C}') := \mathbb{E}_{\mathbf{c}'\sim q_{\mathbf{c}'}}\Big[\big\|\mathcal{E}\big(\boldsymbol{g}(\mathbf{c}')\big) - \mathbf{c}'\big\|_2^2\Big], \tag{9}$$

where $q_{\mathbf{c}'}$ is a distribution with support $\mathcal{C}'$. The loss can be viewed as sampling an OOD combination of concepts $\mathbf{c}'$ by composing inferred in-distribution concepts (i.e. from $\mathcal{E}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$). This synthesized concept is then passed through the decoder to generate an OOD input $\boldsymbol{g}(\mathbf{c}')$. This sample is then re-encoded with $\mathbf{c}' = \mathcal{E}\big(\boldsymbol{g}(\mathbf{c}')\big)$. Finally, the loss regularizes the explainer $\mathcal{E}$ to serve as an approximate inverse of the decoder function for OOD samples. The choice of the compositionality of concepts is crucial and has proved to be stable (Miao et al., 2022).

3.4 Concept Activation Region Discovery and Alignment

Concept Alignment. To enhance interpretability, our framework discovers and aligns learned concepts using techniques inspired by Crabbé & van der Schaar (2022) works, Concept Activation Regions (CAR), better suited to time series tasks as it doesn't assume linear separability. Specifically, we enable the manual definition of low-level concepts, which are subsequently aligned with higher-level abstractions learned by the model. This alignment is performed using kernel-based Support Vector Regression (SVR) and Support Vector Classification (SVC) to distinguish between the activations produced by the network for samples containing the target concept and those for randomly selected samples. We provide the Algorithm in the Appendix B.7.1.

Automated Global Features Interpretation. For interpreting time series, it is practically important to detect the presence or absence of dependence of concepts in certain input time coordinates. To approach this in a more principled probabilistic manner, we introduce Bernoulli random variables for each order interaction and data dimension. Specifically, we introduce $\boldsymbol{m}_k^{(j)} \sim \text{Bernoulli}(\boldsymbol{p}_0)$ for interaction order $k \in \{1, \dots, d\}$, and for each concepts $j \in \{1, \dots, d\}$, where each $\boldsymbol{m}_k^{(j)}$ indicates the presence of a non-zero effect of $\boldsymbol{\psi}_k$ and thus of the concepts. More formally, $\boldsymbol{g}(\mathbf{c})$ follows:

$$\boldsymbol{g}(\mathbf{c}) := \boldsymbol{\psi}_0 + \sum_{j=1}^{d} \boldsymbol{\psi}_1(\boldsymbol{c}_j) + \sum_{j=1}^{d-1} \boldsymbol{\psi}_2(\boldsymbol{c}_j, \boldsymbol{c}_{j+1}) + \sum_{j=1}^{d-2} \boldsymbol{\psi}_3(\boldsymbol{c}_j, \boldsymbol{c}_{j+1}, \boldsymbol{c}_{j+2}) + \cdots + \boldsymbol{\psi}_d(\mathbf{c}),$$

where $\boldsymbol{\psi}_0$ denote the bias term, $\boldsymbol{\psi}_1$ captures the first-order contribution of $\boldsymbol{c}_j$, and $\boldsymbol{\psi}_2$ models the second-order interaction between $\boldsymbol{c}_j$ and $\boldsymbol{c}_{j-1}$, with summation over terms $\boldsymbol{\psi}_k$ for $k = 2, \dots d$. Higher-order terms $\boldsymbol{\psi}_d$ represent interactions across successive concepts $\mathbf{c}$. These terms are elementwise multiplied with their respective Bernoulli sparsity masks $\boldsymbol{m}_k^{(j)}$, activating or deactivating specific interaction effects. This generalizes the neural functional ANOVA decomposition (Märtens & Yau, 2020) and GAM (Yang et al., 2024), offering interpretability by visualizing individual concept impacts and their interactions.

Training Setting. `TimeSAE` minimizes all the previously described loss functions. Notably, we include a *reconstruction term* to ensure *label fidelity* between the prediction made directly from $f(\mathbf{x})$ and the one obtained from the reconstructed input $f\big(\boldsymbol{g}(\mathcal{E}(\mathbf{x}))\big)$. We refer to this term as the *label-fidelity loss*, denoted by $\mathcal{L}_{\text{label-fidelity}}$.

$$\mathcal{L}_{\text{TimeSAE}} = \mathcal{L}_{\text{SAE}} + \mathcal{L}_{\text{label-fidelity}} + \alpha \mathcal{L}_{cc} + \lambda \mathcal{L}_{cf}, \tag{10}$$

where $(\alpha, \lambda) \in \mathbb{R}^2$ are hyperparameters adjusting the losses, and `TimeSAE` is optimized end-to-end.

4 Experimental Setup

Black-Box Models. We employ two types of black-box models: we trained a Transformer-based classifier (Vaswani et al., 2017), DLinear (Zeng et al., 2023), and PatchTS (Nie et al., 2022) models for regression tasks, with hyperparameters carefully tuned to maximize predictive performance. In addition, we incorporate pretrained large-scale models for forecasting and classification: TimeGPT (Garza et al., 2023), accessed via API; Chronos (Ansari et al., 2024), an open-source black-box model with 188 billion parameters designed for regression tasks. Due to space constraints, additional models such as Moments (Goswami et al., 2024), TimeFM (Das et al., 2024), and Informer (Zhou et al., 2021) are reported in the Appendix B.3.1. For all predictors, we verified that the models achieved satisfactory performance on the testing set before the explainability evaluation.

Datasets. We use two synthetic datasets with known ground-truth explanations: **(1) FreqShapes** and **(2) SeqComb-UV** adapted from Queen et al. (2023). Datasets are designed to capture diverse temporal dynamics in both univariate and multivariate settings (we give more details in Section B.1.1). We employ four datasets from real-world time series. For classification tasks, we consider **(1) ECG** dataset (Moody & Mark, 2001) that consists of arrhythmia detection, which includes cardiac disorders with ground-truth explanations defined as the QRS interval (Queen et al., 2023). **(2) PAM** (Reiss & Stricker, 2012) human activity recognition. For regression, **(3) ETTH-1** and **(4) ETTH-2** which are energy demand datasets (Ruhnau et al., 2019). Finally, we introduce **(5) EliteLJ Dataset**, a new real-world sports dataset, consisting of skeletal athlete pose sequences along with the corresponding performance metrics. More details are provided in the Appendix B.1.2.

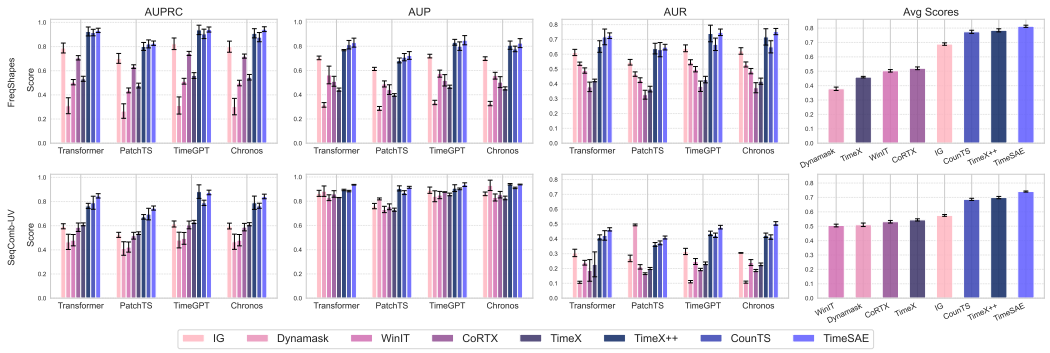

Figure 2: Explanation performance on all datasets and metrics (AUPRC, AUP, AUR). Higher is better. The rightmost panel shows average scores. Methods are ranked left to right from worst to best.

Explanation Evaluation. Given that precise salient features are known, we utilize them as ground truth for evaluating explanations. At each time step, features causing prediction label changes are attributed an explanation of 1, whereas those that do not affect such changes are 0. Following Crabbé & Van Der Schaar (2021), we evaluate the quality of explanations with area under precision (AUP), area under recall (AUR), and also AUPRC, for consistency from Queen et al. (2023), which combines the two. Following TimeX (Liu et al., 2024b), we assess distributional similarity using KL divergence and MMD (Gretton et al., 2012), with smaller values denoting closer alignment. We further estimate the *log-likelihood* of explanations via KDE (Parzen, 1962).

Faithfulness Evaluation. We evaluate *faithfulness* to assess the reliability of our interpretations, following the methodology introduced in Parekh et al. (2022). Faithfulness measures whether the features identified as relevant are truly important for the classifier's predictions. Since "ground-truth" feature importance is rarely available, attribution-based methods typically evaluate faithfulness by removing features (e.g., setting their values to zero) and observing changes in the classifier's output. However, we enable the simulation of feature removal in time-series data by deactivating a set of components. For a sample $\mathbf{x}$ with predicted $\mathbf{y}$, we remove the set of relevant components in $\mathbf{c}$ to obtain $\mathbf{c}^-$ and generate a new instance $\widetilde{\mathbf{x}}^- = \boldsymbol{g}(\mathbf{c}^-)$. The faithfulness score is then computed as: $\mathcal{F}_{\mathbf{x}} = ||f(\mathbf{x}) - f(\widetilde{\mathbf{x}}^-)||_2^2$ where $f(\mathbf{x})$ and $f(\widetilde{\mathbf{x}}^-)$ represent the model's output before and after concept removal. A higher $\mathcal{F}_{\mathbf{x}}$ indicates a greater impact of the removed components on the output, supporting the faithfulness of the interpretation.

Baselines details. The proposed method `TimeSAE` is evaluated against eight state-of-the-art explainability methods: Integrated Gradients (IG) (Sundararajan et al., 2017), Dynamask (Crabbé & Van Der Schaar, 2021), WinIT (Leung et al., 2023), CoRTX (Chuang et al., 2023), SGTGRAD (Ismail et al., 2021), TimeX (Queen et al., 2023), TimeX++ (Liu et al., 2024b), and CounTS (Yan & Wang, 2023). Further implementation details are provided in the Appendix B.4.

5 Results and Discussions

5.1 Synthetic and Real-world Datasets

Figure 2 and Figure 3 report the performance of different explanation methods on univariate and multivariate datasets, presented as the mean and standard deviation over 10 random seeds (applies to all subsequent tables/figures). Our approach consistently achieves the best results across metrics (AUPRC, AUP, and AUR). The performance metric (AUPRC) on real datasets, illustrated in Figure 3, show that `TimeSAE` surpasses existing methods, including TimeX++, across all datasets. We perform paired t-tests at the 5% significance level to evaluate whether `TimeSAE` significantly outperforms other methods. The results indicate that `TimeSAE` achieves statistically the highest performance on several datasets (e.g., ECG and PAM), while other models like TimeX++ and CounTS remain competitive.

In terms of faithfulness, Table 1 shows results of explanation methods applied to different black-box models, and for multiple datasets. Our `TimeSAE` consistently reaches more faithful results than

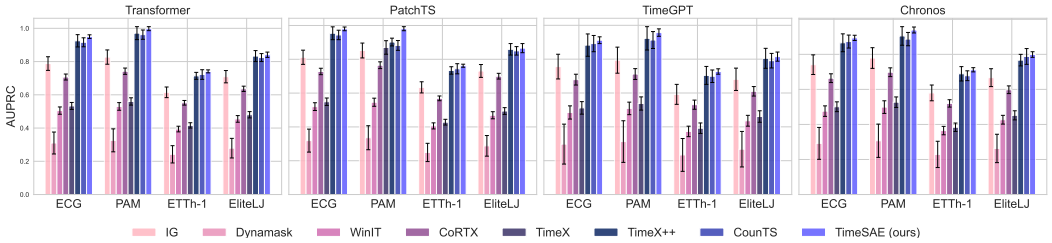

Figure 3: AUPRC explanation performance (higher is better) across methods for each dataset.

Table 1: The Faithfulness $\mathcal{F}_{\mathbf{x}}$ metric performance across classification (†) and regression (‡) tasks with different datasets. For all metrics, higher values are better, and the colors represent the top **Top-1**, Top-2, and Top-3 rankings.

Black-Box → / Method ↓	Transformer ECG†	Transformer PAM†	PatchTS ETTH-1‡	PatchTS ETTH-2‡	DLinear EliteLJ‡	Rank
IG	0.92±0.102	0.89±0.104	1.00±0.107	0.95±0.101	0.91±0.109	9.0
Dynamask	1.05±0.099	1.00±0.095	1.15±0.096	1.12±0.093	1.04±0.097	8.0
WinIT	1.10±0.095	1.08±0.092	1.18±0.091	1.17±0.090	1.06±0.093	6.9
CoRTX	1.15±0.088	1.10±0.087	1.22±0.090	1.20±0.088	1.14±0.089	5.6
TimeX	1.10±0.083	1.22±0.091	1.35±0.090	1.28±0.089	1.10±0.094	5.5
TimeX++	1.65±0.097	1.58±0.088	1.75±0.084	1.70±0.086	1.44±0.087	3.6
CounTS	**1.86±0.075**	2.05±0.074	1.60±0.080	1.50±0.081	1.89±0.071	2.3
TimeSAE (ours)	1.78±0.078	**2.15±0.080**	**2.12±0.072**	**2.09±0.069**	**1.86±0.032**	**1.7**

other baselines, highlighted by its best ranking. We further investigate the faithfulness and its link with counterfactuals from Theorem 1 in our ablation study below.

5.2 Ablation Study

Concepts Consistency in OOD. To assess the effectiveness of the Concepts Consistency in out-of-distribution (OOD) generalization, we evaluate the model's ability to generate explanations that remain faithful and robust when exposed to OOD data, i.e., tested on ETTh-2 while TimeSAE is trained on Etth-1, widely used as OOD generalization benchmark. These datasets are collected from different countries, exhibit distinct seasonal and frequency patterns, and thus serve as a suitable testbed for OOD evaluation (Liu et al., 2024a). This OOD setting is given in Table 2-a. We report in Table 2-b the combined results of KDE, KL, MMD, AUPRC and $\mathcal{F}_{\mathbf{x}}$, comparing in-domain and cross-domain settings. These results show that our proposed TimeSAE not only achieves higher predictive performance (AUPRC) but also produces more faithful and distributionally aligned explanations (lower KDE shift, KL divergence, and MMD).

Effectiveness of Counterfactual for Faithful Explanations. We evaluate the contribution of counterfactual supervision in enhancing explanation faithfulness by evaluating the proposed framework using the $\mathcal{F}_{\mathbf{x}}$ score across five diverse time series datasets and multiple black-box predictors. As shown in Figure 4-b, we examine three variants of the TimeSAE model: one trained with the full objective Equation (10), another without the counterfactual loss term $\mathcal{L}_{cf}$, and a third version trained

Table 2: Distributional alignment and performance evaluation for in-distribution (ID) and out-of-distribution (OOD) settings. (a) Dataset definition and statistics. (b) Evaluation of method performance. Higher is better for KDE, AUPRC, and $\mathcal{F}_{\mathbf{x}}$; lower is better for KL and MMD.

(a) Setting & Statistics

Definition & Statistics			
Dataset Pair	**KDE**	**KL**	**MMD**
● ID: ETTh1 → ETTh1-val trained on ETTh1, and tested on ETTh1-val	-0.110 ± 0.01	0.039 ± 0.002	0.014 ± 0.001
○ OOD: ETTh1 → ETTh2 trained on ETTh1, and tested on ETTh2	-0.295 ± 0.02	0.421 ± 0.03	0.161 ± 0.01

(b) Evaluation

Method	Setting	KDE ↑	KL ↓	MMD ↓	AUPRC ↑	$\mathcal{F}_{\mathbf{x}}$ ↑
IG	● ID	-46.10$_{\pm 1.3}$	0.295$_{\pm 0.025}$	0.027$_{\pm 0.004}$	0.422$_{\pm 0.045}$	1.38$_{\pm 0.06}$
	○ OOD	-49.45$_{\pm 1.6}$	0.355$_{\pm 0.032}$	0.120$_{\pm 0.012}$	0.394$_{\pm 0.03}$	1.31$_{\pm 0.07}$
TimeX	● ID	-45.30$_{\pm 1.2}$	0.288$_{\pm 0.02}$	0.024$_{\pm 0.003}$	0.416$_{\pm 0.04}$	1.35$_{\pm 0.05}$
	○ OOD	-50.82$_{\pm 1.5}$	0.342$_{\pm 0.03}$	0.115$_{\pm 0.01}$	0.401$_{\pm 0.035}$	1.28$_{\pm 0.06}$
TimeX++	● ID	-44.12$_{\pm 1.1}$	0.198$_{\pm 0.02}$	0.019$_{\pm 0.002}$	0.714$_{\pm 0.05}$	1.75$_{\pm 0.07}$
	○ OOD	-48.77$_{\pm 1.3}$	0.265$_{\pm 0.03}$	0.101$_{\pm 0.01}$	0.622$_{\pm 0.04}$	1.70$_{\pm 0.08}$
TimeSAE (Ours)	● ID	-43.55$_{\pm 1.1}$	0.182$_{\pm 0.01}$	0.016$_{\pm 0.002}$	0.741$_{\pm 0.05}$	2.12$_{\pm 0.05}$
	○ OOD	**-47.21**$_{\pm 1.3}$	**0.245**$_{\pm 0.02}$	**0.089**$_{\pm 0.01}$	**0.641**$_{\pm 0.03}$	**2.09**$_{\pm 0.06}$

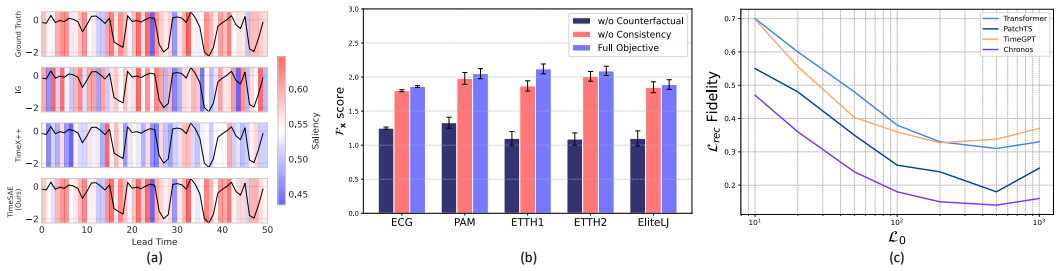

Figure 4: **(a)** Examples of explanation compared to the ground truth on the FreqShapes dataset. **(b)** Effects of excluding the *Concepts Consistency* and *Counterfactual* term on the Faithfulness metric $\mathcal{F}_{\mathbf{x}}$, with standard deviations shown over 10 runs. Using counterfactuals produces more faithful explanations. **(c)** Effect of sparsity on reconstruction fidelity. Increasing sparsity generally improves interpretability; however, excessive sparsity can compromise fidelity.

without the consistency loss $\mathcal{L}_{cc}$. The results consistently show that incorporating counterfactual objectives significantly improves faithfulness across all settings. Notably, models trained with $\mathcal{L}_{cf}$, especially when combined with $\mathcal{L}_{cc}$, yield higher $\mathcal{F}_{\mathbf{x}}$ scores, demonstrating that counterfactual signals help guide the model toward learning more faithful and semantically meaningful explanations. These improvements hold across datasets and predictor types, showing the generalizability of our approach. Qualitative comparisons in Figure 4-a of `TimeSAE` 's explanations with the ground truth against TimeX++ and IG show that `TimeSAE` better captures attributions to temporal features.

Trade-off Between Sparsity and Reconstruction. Although sparsity can improve the monosemanticity of concepts (Pach et al., 2025), forcing sparsity also leads to *lower fidelity* of reconstruction. In Figure 4-c, we evaluate reconstruction error as sparsity increases (measured by $\mathcal{L}_0$). The results show that, when the representations become overly sparse (low $\mathcal{L}_0$), a clear drop in reconstruction fidelity (higher $\mathcal{L}_{\text{rec-fidelity}}$) is observed. We further extend this analysis to the TopK sparsity mechanism discussed in Section 3, referred to as TimeSAE-TopK, with results reported in Appendix B. The results underscore JumpReLU's ability to adapt seamlessly to data, whereas TopK is more sensitive to hyperparameter choices.

6 Conclusion and Future Work

We address the challenge of post-hoc explainability for time-series black-box models by employing sparse autoencoders and exploring the causal relationships between concepts within explanation-embedded instances to ensure faithful explanations. Our proposed approach, `TimeSAE` , automates dictionary learning, generating time series explanations that preserve labels and are consistent with the original data distribution. Through experiments on both synthetic and real-world datasets, we show that `TimeSAE` outperforms current explainability methods in terms of faithfulness and robustness to distribution shift, with successful applications in fields such as energy forecasting and sports analytics. In particular, our results highlight `TimeSAE` 's ability to accurately capture the complex dynamics of pre-trained time series models. This work emphasizes the importance of concept composition and causality in producing high-quality explanations. Future investigations could focus on constructing white-box models based on these concepts and exploring the interpretability of layers within black-box models.

Limitations. While `TimeSAE` produces faithful and interpretable explanations, it requires a sufficiently large and representative dataset for effective training, which may limit its applicability in data-scarce domains. Additionally, the performance of the model is sensitive to hyperparameter settings such as sparsity levels and dictionary size, requiring careful tuning to maintain robustness and generalizability across different tasks. A discussion of these limitations can be found in the Appendix C.

References

Tomiwa Sunday Adebayo, Abraham Ayobamiji Awosusi, Dervis Kirikkaleli, Gbenga Daniel Akinsola, and Madhy Nyota Mwamba. Can CO2 emissions and energy consumption determine the economic

performance of south korea? A time series analysis. *Environmental Science and Pollution Research*, pp. 38969–38984, 2021.

David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

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

Omar Bahri, Peiyu Li, Soukaina Filali Boubrahimi, and Shah Muhammad Hamdi. Discord-based counterfactual explanations for time series classification. *Data Mining and Knowledge Discovery*, 38(6):3347–3371, 2024.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *SIGKDD*, pp. 2565–2573, 2021.

Jack Brady, Roland S. Zimmermann, Yash Sharma, Bernhard Schölkopf, Julius Von Kügelgen, and Wieland Brendel. Provably learning object-centric representations. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3038–3062. PMLR, 23–29 Jul 2023.

Yu-Neng Chuang, Guanchu Wang, Fan Yang, Quan Zhou, Pushkar Tripathi, Xuanting Cai, and Xia Hu. CoRTX: Contrastive framework for real-time explanation. In *ICLR*, pp. 1–23, 2023.

Jonathan Crabbé and Mihaela Van Der Schaar. Explaining time series predictions with dynamic masks. In *ICML*, pp. 2166–2177, 2021.

Jonathan Crabbé and Mihaela van der Schaar. Concept activation regions: A generalized framework for concept-based explanations. *Advances in Neural Information Processing Systems*, 35:2590–2607, 2022.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Abdelkader Dairi, Fouzi Harrou, Abdelhafid Zeroual, Mohamad Mazen Hittawe, and Ying Sun. Comparative study of machine learning methods for covid-19 transmission forecasting. *Journal of biomedical informatics*, 118:103791, 2021.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

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

Cherrelle Eid, Paul Codani, Yannick Perez, Javier Reneses, and Rudi Hakvoort. Managing electric flexibility from distributed energy resources: A review of incentives for market design. *Renewable and Sustainable Energy Reviews*, 64:237–247, 2016.

Joseph Enguehard. Learning perturbations to explain time series predictions. In *ICML*, pp. 9329–9342, 2023.

Qi Gan, Mounîm A El-Yacoubi, Eric Fenaux, Stéphan Clémençon, et al. Human pose estimation based biomechanical feature extraction for long jumps. In *2024 16th International Conference on Human System Interaction (HSI)*, pp. 1–6. IEEE, 2024.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Yair Gat, Nitay Calderon, Amir Feder, Alexander Chapanin, Amit Sharma, and Roi Reichart. Faithful explanations of black-box nlp models using llm-generated counterfactuals. *arXiv preprint arXiv:2310.00603*, 2023.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *CoRR*, abs/1907.07165, 2019. URL http://arxiv.org/abs/1907.07165.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

Aya Abdelsalam Ismail, Hector Corrada Bravo, and Soheil Feizi. Improving deep learning interpretability by saliency guided training. In *NeurIPS*, pp. 26726–26739, 2021.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural networks*, 116:237–245, 2019.

Shruti Kaushik, Abhinav Choudhury, Pankaj Kumar Sheron, Nataraj Dasgupta, Sayee Natarajan, Larry A Pickett, and Varun Dutt. AI in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in Big Data*, 3:4, 2020.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

Kin Kwan Leung, Clayton Rooke, Jonathan Smith, Saba Zuberi, and Maksims Volkovs. Temporal dependencies in feature importance for time series prediction. In *ICLR*, pp. 1–18, 2023.

Peiyu Li, Omar Bahri, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. Cels: Counterfactual explanations for time series data via learned saliency maps. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 718–727. IEEE, 2023.

Peiyu Li, Pouya Hosseinzadeh, Omar Bahri, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. Reliable time series counterfactual explanations guided by shapedba. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 1574–1579. IEEE, 2024.

Haoxin Liu, Harshavardhan Kamarthi, Lingkai Kong, Zhiyuan Zhao, Chao Zhang, and B Aditya Prakash. Time-series forecasting for out-of-distribution generalization using invariant learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 31312–31325, 2024a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Zichuan Liu, Tianchun Wang, Jimeng Shi, Xu Zheng, Zhuomin Chen, Lei Song, Wenqian Dong, Jayantha Obeysekera, Farhad Shirani, and Dongsheng Luo. Timex++: Learning time-series explanations with information bottleneck. *arXiv preprint arXiv:2405.09308*, 2024b.

Zichuan Liu, Yingying Zhang, Tianchun Wang, Zefan Wang, Dongsheng Luo, Mengnan Du, Min Wu, Yi Wang, Chunlin Chen, Lunting Fan, and Qingsong Wen. Explaining time series via contrastive and locally sparse perturbations. In *ICLR*, pp. 1–21, 2024c.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Kaspar Märtens and Christopher Yau. Neural decomposition: Functional anova with variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pp. 2917–2927. PMLR, 2020.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *ICML*, pp. 15524–15543, 2022.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, 20(3):45–50, 2001.

Andrew Ng. Sparse autoencoder. http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf, 2011. CS294A Lecture notes.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. *arXiv preprint arXiv:2304.06129*, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *arXiv preprint arXiv:2504.02821*, 2025.

Jayneel Parekh, Pavlo Mozharovskyi, and Florence d'Alché-Buc. A framework to learn with interpretation. *Advances in Neural Information Processing Systems*, 34:24273–24285, 2021.

Jayneel Parekh, Sanjeel Parekh, Pavlo Mozharovskyi, Florence d'Alché-Buc, and Gaël Richard. Listen to interpret: Post-hoc interpretability for audio networks with nmf. *Advances in Neural Information Processing Systems*, 35:35270–35283, 2022.

Jayneel Parekh, Quentin Bouniot, Pavlo Mozharovskyi, Alasdair Newson, and Florence d'Alché Buc. Restyling unsupervised concept based interpretable networks with generative models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CexatBp6rx.

Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Owen Queen, Thomas Hartvigsen, Teddy Koker, Huan He, Theodoros Tsiligkaridis, and Marinka Zitnik. Encoding time-series explanations through self-supervised model behavior consistency. In *NeurIPS*, 2023.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, Janos Kramar, Rohin Shah, and Neel Nanda. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *ISWC*, pp. 108–109, 2012.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *SIGKDD*, pp. 1135–1144, 2016.

Oliver Ruhnau, Lion Hirth, and Aaron Praktiknjo. Time series of heat demand and heat pump efficiency for energy system modeling. *Scientific data*, 6(1):1–10, 2019.

Anirban Sarkar, Deepak Vijaykeerthy, Anindya Sarkar, and Vineeth N Balasubramanian. A framework for learning ante-hoc explainable models via concepts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10286–10295, 2022.

Jimeng Shi, Vitalii Stebliankin, and Giri Narasimhan. The power of explainability in forecast-informed deep learning models for flood mitigation. *arXiv preprint arXiv:2310.19166*, 2023.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, pp. 3319–3328, 2017.

Harini Suresh, Nathan Hunt, Alistair Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding with deep neural networks. In *MLHC*, pp. 322–337, 2017.

Sana Tonekaboni, Shalmali Joshi, Kieran Campbell, David K Duvenaud, and Anna Goldenberg. What went wrong and when? Instance-wise feature importance for time-series black-box models. In *NeurIPS*, pp. 799–809, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Zhendong Wang, Ioanna Miliou, Isak Samsten, and Panagiotis Papapetrou. Counterfactual explanations for time series forecasting. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 1391–1396. IEEE, 2023.

Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable compositional generalization for object-centric learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35:38571–38584, 2022.

Jingquan Yan and Hao Wang. Self-interpretable time series prediction with counterfactual explanations. In *International Conference on Machine Learning*, pp. 39110–39125. PMLR, 2023.

Linxiao Yang, Yunze Tong, Xinyue Gu, and Liang Sun. Explain temporal black-box models via functional decomposition. In *Forty-first International Conference on Machine Learning*, 2024.

Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956, 2009.

Zeyu Yun, Yubei Chen, Bruno A Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. *arXiv preprint arXiv:2103.15949*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 11106–11115, 2021.

Supplementary Material

Table of Contents

Symbol	Description
C	Number of features for time series
T	Denote the sequence length
d_x	The dimension $C \times T$, where T is time and C features.
d_z	The dimension $r \cdot C \times T$, where r is the latent Dimensionality Ratio.
$\mathbf{x} \in \mathbb{R}^{d_x}$	Observation
$\mathbf{x}^{cf} \in \mathbb{R}^{d_x}$	The counterfactual Observation
$\mathbf{y} \in \mathbb{R}^{d_y}$	Ground-truth
$\mathbf{y}^{cf} \in \mathbb{R}^{d_y}$	The counterfactual ground-truth
$\boldsymbol{c} \in \mathbb{R}^{d_z}$	Vector of concepts factors
$\mathcal{C} \subseteq \mathbb{R}^{d_z}$	Support of concepts $\boldsymbol{c}$
I_k	Intervention on the concepts $\boldsymbol{c}_k$
$\mathcal{E}$	Explainer or encoder function
$\boldsymbol{g}$	Decoder function
f	The Blackbox model to explain
$\lvert\mathcal{D}\rvert$	The cardinal of the dataset $\mathcal{D}$

Table 3: Table of Notation.

A Definitions, Assumptions, and Proofs

This section provides the formal definitions, assumptions, and theoretical justifications underpinning the methods discussed in the main text. We introduce key concepts related to counterfactual explanations, specifically the Causal Concept Effect (CaCE) and its approximation in a model-agnostic setting using explainers Section A.2. We then present a formal proof of faithfulness for the proposed Approximate Counterfactual Explanation method, particularly within the framework of Sparse Autoencoders.

A.1 Approximated Counterfactual Explanation

Causal effects and model explanations are naturally linked to counterfactuals (CFs), as they quantify how model outputs change under hypothetical interventions. However, computing exact counterfactuals often requires full knowledge of the underlying data-generating process, which is typically unavailable. To address this, we adopt an *approximated* counterfactual approach, leveraging only the explainer $\mathcal{E}$ and observed data.

Definition 4 (**CaCE** (Goyal et al., 2019))**.** *Given an intervention $I_k : \boldsymbol{c}_k \mapsto \boldsymbol{c}'_k$, the CaCE is:*

$$\begin{aligned}\text{CaCE}_f(I_k, \boldsymbol{c}_k, \boldsymbol{c}'_k) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[f(\mathbf{x})|do(I_k = \boldsymbol{c}'_k)\right] \\ - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[f(\mathbf{x})|do(I_k = \boldsymbol{c}_k)\right].\end{aligned} \tag{11}$$

By using approximated counterfactuals, we can efficiently estimate the causal effect of concepts on model predictions in a model-agnostic way. This allows us to generate explanations that capture the influence of individual concepts while remaining computationally tractable, even in complex, high-dimensional datasets.

Definition 5 (**Approximated Counterfactual Explanation (Gat et al., 2023)**)**.** *Given a dataset $\mathcal{D}$, an explainer $\mathcal{E}$, and an intervention $I_k : \boldsymbol{c}_k \mapsto \boldsymbol{c}'_k$, the approximated counterfactual explanation $\mathcal{E}^{cf}$ is defined as:*

$$\mathcal{E}^{cf} = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathcal{E}(\tilde{\mathbf{x}}_{\boldsymbol{c}'_k}) - \mathcal{E}(\tilde{\mathbf{x}}_{\boldsymbol{c}_k}), \tag{12}$$

where $\tilde{\mathbf{x}}_{\boldsymbol{c}'_k}$ denotes the explanation-embedded instance after the intervention, and $\tilde{\mathbf{x}}_{\boldsymbol{c}_k}$ before the intervention.

A.2 Proof of Faithfulness for Approximate Counterfactuals in Sparse Autoencoders

Theorem 1 (**Faithfulness in Sparse Autoencoder-Based Approximate Counterfactuals**)**.** *Let $\mathbf{x}$ be a time-series input and f a black-box model whose true output is $\mathbf{y} = f(\mathbf{x})$. Suppose $(\mathcal{E}, \boldsymbol{g})$ is an encoder-decoder, where $\mathcal{E}$ encodes $\mathbf{x}$ to latent concepts, and $\boldsymbol{g}$ decodes these concepts into $\widetilde{\mathbf{x}} = \boldsymbol{g}(\mathcal{E}(\mathbf{x}))$ such that $\forall \mathbf{x} \in \mathcal{D}$, $f(\widetilde{\mathbf{x}}) \approx f(\mathbf{x})$. For an intervention, define an* approximate counterfactual *by altering concepts $\mathbf{c} \mapsto \mathbf{c}^{cf}$, and let $\widetilde{\mathbf{x}}^{cf} = \boldsymbol{g}\big(\mathbf{c}^{cf}\big)$. Assume that*

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[\,|\,f(\widetilde{\mathbf{x}}^{cf}) - \mathbf{y}^{cf}\,|\,\big] \;\leq\; \epsilon_{cf}, \tag{5}$$

where $\mathbf{y}^{cf}$ is the "true" counterfactual label (i.e., what $f(\mathbf{x})$ would be under the exact causal intervention), and ε_{cf} is a small approximation error. Then, for any pair of interventions $I_1 : c_1 \mapsto c'_1$ and $I_2 : c_2 \mapsto c'_2$, if the true *causal effects satisfy*

$$\texttt{CaCE}_f(I_1, c_1, c'_1) \;>\; \texttt{CaCE}_f(I_2, c_2, c'_2), \tag{6}$$

there exists a sufficiently small ϵ_{cf} so that

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f(\widetilde{\mathbf{x}}^{cf}_{I_1}) - f(\widetilde{\mathbf{x}})\Big] \;>\; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f(\widetilde{\mathbf{x}}^{cf}_{I_2}) - f(\widetilde{\mathbf{x}})\Big], \tag{7}$$

where $\widetilde{\mathbf{x}}^{cf}_{I_1}$ and $\widetilde{\mathbf{x}}^{cf}_{I_2}$ are the explanation-embedded instances respectively obtained after interventions I_1 and I_2. This preserves the ordering *of causal effects, i.e. order faithfulness.*

Proof. To establish order-faithfulness, we aim to show that the Sparse Autoencoder-Based Approximate Counterfactuals preserve the ordering of causal effects as dictated by the black-box model f. Specifically, if intervention I_1 has a greater true causal effect than intervention I_2, then the expected change in f's output when applying I_1 should exceed that of I_2 in the approximate counterfactuals generated by the autoencoder.

Step 1. True vs. Approximate Counterfactual Effects. Denote the *true* causal effect of an intervention I by

$$\delta_{\text{true}}(I) \;=\; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f\big(\mathbf{x}_I^{cf}\big) \;-\; f(\mathbf{x})\Big],$$

where $\mathbf{x}_I^{cf}$ is perfectly causal version of $\mathbf{x}$ under I. By hypothesis, for two interventions I_1 and I_2 we have $\delta_{\text{true}}(I_1) > \delta_{\text{true}}(I_2)$.

Define the *approximate* effect as

$$\delta_{\text{approx}}(I) \;=\; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Big[f\big(\widetilde{\mathbf{x}}_I^{cf}\big) \;-\; f\big(\widetilde{\mathbf{x}}\big)\Big],$$

where $\widetilde{\mathbf{x}}_I^{cf} = \boldsymbol{g}\big(\mathcal{E}(\mathbf{x})_I^{cf}\big)$ is obtained by modifying the latent encoding of $\mathbf{x}$ to reflect the intervention I.

Step 2. Bounding the Approximation Error. By assumption,

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[\big|f\big(\widetilde{\mathbf{x}}_I^{cf}\big) \;-\; f\big(\mathbf{x}_I^{cf}\big)\big|\big] \;\le\; \varepsilon_{\text{cf}},$$

and also $f(\widetilde{\mathbf{x}}) \approx f(\mathbf{x})$ implies a small *reconstruction* error ε_{rec}. Combining these,

$$\big|\delta_{\text{approx}}(I) - \delta_{\text{true}}(I)\big| \;\le\; \varepsilon_{\text{cf}} + \varepsilon_{\text{rec}}.$$

Since $\delta_{\text{true}}(I_1)$ is strictly larger than $\delta_{\text{true}}(I_2)$, there is a positive gap

$$\delta \;=\; \delta_{\text{true}}(I_1) \;-\; \delta_{\text{true}}(I_2) \;>\; 0.$$

Choosing $\varepsilon_{\text{cf}} + \varepsilon_{\text{rec}} < \delta/2$ prevents the approximate effects from inverting this gap. Formally,

$$\delta_{\text{approx}}(I_1) - \delta_{\text{approx}}(I_2) \;=\; \big[\delta_{\text{true}}(I_1) + \epsilon_1\big] \;-\; \big[\delta_{\text{true}}(I_2) + \epsilon_2\big] \;=\; \delta + (\epsilon_1 - \epsilon_2),$$

where $|\epsilon_i| \le \varepsilon_{\text{cf}} + \varepsilon_{\text{rec}}$. Thus,

$$\delta + (\epsilon_1 - \epsilon_2) \;\ge\; \delta - 2(\varepsilon_{\text{cf}} + \varepsilon_{\text{rec}}).$$

If $\varepsilon_{\text{cf}} + \varepsilon_{\text{rec}} < \delta/2$, then this quantity remains positive, ensuring

$$\delta_{\text{approx}}(I_1) \;>\; \delta_{\text{approx}}(I_2).$$

Hence, for sufficiently small ε_{cf} (and reconstruction error), the approximate counterfactual preserves the true ordering of the causal effects in expectation over $\mathcal{D}$. This completes the proof. □

B Experimental Setting and Additional Experiments

B.1 Dataset

In this work, we use multiple time series datasets across different case studies. We rely primarily on publicly available datasets released under the MIT License, ensuring unrestricted access for research purposes. In addition, we generate a synthetic dataset using the scripts provided by Queen et al. (2023). To further support evaluation of explainability and reasoning in our proposed `TimeSAE` framework, we also introduce a new real-world dataset, *EliteLJ*, which contains human pose sequences specifically collected for this study.

B.1.1 Synthetic Dataset

To ensure consistent evaluation and enable direct comparison with existing methods, we adopt the synthetic dataset design introduced by Queen et al. (2023), which provides controlled settings for analyzing time series explanation capabilities. This benchmark suite consists of carefully structured datasets that isolate specific temporal properties, enabling ground-truth explanations.

FreqShapes. This dataset tests the ability to detect periodic anomalies based on both shape and frequency. Each sample contains recurring spike patterns, either upward or downward, occurring at regular intervals. Two frequencies (10 and 17 time steps) and two spike shapes are combined to create four distinct classes: (0) downward spikes every 10 steps, (1) upward spikes every 10 steps, (2) downward spikes every 17 steps, and (3) upward spikes every 17 steps. The explanatory signal lies in both the spike occurrence and their periodicity, with labeled explanation regions marking the spike positions.

SeqComb-UV. This univariate dataset focuses on recognizing ordered shape patterns. Time series are injected with two non-overlapping subsequences exhibiting either increasing or decreasing trends. Each subsequence is 10–20 time steps long and shaped using sinusoidal signals with variable wavelengths. Four classes are formed based on the arrangement: class 0 contains no signal (null baseline), class 1 contains two increasing patterns (I, I), class 2 contains two decreasing ones (D, D), and class 3 has an increasing followed by a decreasing trend (I, D). The predictive cues reside in the subsequence configurations, which are used as ground-truth explanation masks.

Additional Synthetic datasets. We also include two more challenging datasets from Queen et al. (2023) (SeqComb-MV and LowVar) designed to test multivariate reasoning and detection of low-variance patterns. This multivariate extension of SeqComb-UV retains the same class structure but distributes the increasing and decreasing patterns across different randomly selected channels. Models must identify not only the temporal location but also the specific sensor responsible for the predictive subsequences. Ground-truth explanations correspond to both the time intervals and channels of the patterns. For LowVar, the predictive signal is a region of low variance within an otherwise noisy multivariate time series. Each class corresponds to the mean value and channel of this low-variance segment. Unlike other datasets, the informative region is subtle, with no sharp change points, making detection more difficult. Ground-truth explanations highlight the location and variable of the low-variance segment.

B.1.2 Real-World Datasets

ECG. A univariate time series dataset of electrocardiogram signals from the UCR archive (Dau et al., 2019), used for anomaly and pattern detection tasks.

ETT (Electricity Transformer Temperature) The ETT[2] dataset is a key dataset used in forecasting benchmarks. It contains two years of data collected from two different counties. To facilitate the analysis of explanation methods, the dataset is divided into two subsets: ETTh1 and ETTh2, which provide hourly data. Each time step includes the target variable "oil temperature" along with six auxiliary power load features (i.e., $C = 6$). The data is partitioned into training, validation, and test sets following a 12:4:4-month split.

[2]Available at: https://github.com/zhouhaoyi/ETDataset

PAM. Physical Activity Monitoring dataset, consisting of multivariate sensor recordings of human motion across various labeled activities.

EliteLJ (Proposed). We collected real-world human pose sequence data from elite long jump competitions to further evaluate our approach. The dataset includes 386 successful long jump attempts recorded during the men's finals of the World Championships, Olympic Games, and European Championships. All videos were publicly available online and recorded at 25 frames per second. To ensure temporal alignment across samples, each jump sequence was clipped to 50 frames (2 seconds), with the take-off from the jump board consistently aligned to frame 26. Notice that in long jump competition videos, each frame contains only one athlete. We used ViTPose (Xu et al., 2022) to estimate 2D skeletal poses of the athletes from each video and applied manual corrections using an online annotation tool provided in Gan et al. (2024). The resulting poses follow the same format as the Human3.6M dataset (Ionescu et al., 2013), with 17 keypoints per frame. Consequently, each pose sequence is represented as a 50 time-step, 34-dimensional time series. The official jump distance for each attempt was used as the ground-truth label. The dataset is released publicly in the project link.

Table 4: Summary of datasets used in our experiments. T: sequence length, D: number of features.

Dataset	#Samples	Train	Val	Test	T	D	Task	Type
FreqShapes	5,000	3,000	1,000	1,000	50	5	Classification	Univariate
SeqComb-UV	6,000	3,600	1,200	1,200	60	10	Classification	Mutivariate
ECG	3,000	2,000	500	500	140	1	Classification	Univariate
ETTh1	8,640	6,000	1,320	1,320	96	7	Regression	Mutivariate
ETTh2	8,640	6,000	1,320	1,320	96	7	Regression	Mutivariate
PAM	5,400	3,500	950	950	100	8	Classification	Mutivariate
EliteLJ	386	270	58	58	50	34	Regression	Mutivariate

B.2 METRICS

We evaluate our TimeSAE methods using three established metrics from Crabbé & Van Der Schaar (2021) that assess feature importance detection as a binary classification problem.

Area Under Precision (AUP) This metric quantifies how accurately our method identifies salient features without generating excessive false positives. AUP integrates precision performance across all possible detection thresholds, measuring the method's specificity in saliency detection.

Area Under Recall (AUR) This metric measures our method's ability to comprehensively capture all truly important features. AUR integrates recall performance across the full threshold range, indicating the method's sensitivity in identifying relevant features.

Area Under Precision-Recall Curve (AUPRC) Following Crabbé & Van Der Schaar (2021); Queen et al. (2023), we employ AUPRC as a unified assessment metric that combines precision and recall information into a single score, providing a balanced evaluation of overall saliency detection performance.

Evaluation Implementation Our evaluation converts the continuous saliency masks produced by TimeSAE methods into binary predictions for comparison against ground truth annotations. Our TimeSAE variants generate continuous saliency, where higher values indicate greater feature importance. We convert these into binary saliency maps through thresholding: features with mask values above threshold τ are classified as salient, while others are deemed non-salient. We compare these binary predictions against ground truth saliency matrices that indicate which features are truly important. By varying the threshold of activation across its full range, we generate precision and recall curves that capture the trade-off between correctly identifying salient features and avoiding false positives. The areas under these curves provide our final AUP, AUR, and AUPRC scores.

B.3 Trained and Large Pretrained Black-Box Models.

In this section, we provide further details on the black-box models used in our experimental framework, as introduced in Section 4. We distinguish between two categories (see Table 5): **(1) Trained Models.** These models are trained from scratch using full access to the dataset. The training follows standard supervised learning procedures, and the resulting models serve as black-box predictors for downstream explanation tasks. **(2) Large Pretrained and Fine-Tuned Models.** When publicly available checkpoints are provided, we optionally fine-tune these models to better adapt to the specific data distribution. This setup allows us to evaluate the generalizability and adaptability of our explanation methods across both domain-specific and foundation-style models.

B.3.1 Trained Black boxes Models

Table 5: Black-Box Models categories.

Model Name	Trained Model	Large Pretrained	Large Finetuned
Transformers	✓	✗	✗
PatchTS	✓	✗	✗
Informer	✓	✗	✗
IFormer	✓	✗	✗
LSTM	✓	✗	✗
Chronos	✗	✓	✗
TimeGPT	✗	✓	✗
TimeFM	✗	✓	✗
Moments	✗	✓	✓
Morai	✗	✓	✓
Moments	✗	✓	✓
TimeGPT	✗	✓	✗

Transformers Originally introduced in NLP (Vaswani et al., 2017), Transformers have been successfully adapted for time series forecasting due to their powerful self-attention mechanism, which can capture long-range temporal dependencies without recurrence. To ensure a fair comparison with the results of Queen et al. (2023) in terms of explanations provided for the Transformer, we adopt the same vanilla Transformer architecture used by the authors. We recall that explanations are provided for the bes performing predictor at test time, as this validation step is essential, as highlighted by Queen et al. (2023).

PatchTS PatchTS (Ghandeharioun et al., 2024) improves Transformer efficiency by dividing the input time series into patches and applying self-attention locally within each patch. This approach reduces computational complexity and enables the model to capture fine-grained temporal patterns, as the patching is performed on the input sequence before being passed to the attention block. We follow the standard implementation of PatchTS/62 available in `https://github.com/yuqinie98/PatchTST`, and our training results achieve a similar MSE and MAE to those reported in the original work (Ghandeharioun et al., 2024). Specifically, our results show a 19.31% reduction in MSE and a 16.1% reduction in MAE, while the main paper reports an overall 21.0% reduction in MSE and 16.7% reduction in MAE.

Informer Informer (Zhou et al., 2021) is designed for long sequence time series forecasting. It introduces a ProbSparse self-attention mechanism that reduces the quadratic complexity of vanilla Transformers to near-linear, enabling the model to handle long sequences. We follow the same training procedure as described in Zhou et al. (2021), specifically outlined in the Hyperparameter Tuning section.

iTransformer Liu et al. (2023) extends Transformer by modeling input-level interactions explicitly through enhanced attention mechanisms, improving multivariate time series forecasting accuracy. Unlike the vanilla Transformer, iTransformer embeds each time series independently into a variate token, allowing the attention module to capture multivariate correlations, while the feed-forward network encodes the individual series representations. This architectural enhancement over the vanilla Transformer may offer valuable insights, and we believe that providing explanations for its behavior could be of interest to the time series community.

LSTM Long Short-Term Memory networks (Karim et al., 2019) remain a baseline for time series due to their ability to retain long-term memory. Though older than Transformers, LSTMs are parameter-efficient, often with fewer than 5 million parameters for moderate-sized problems, and continue to be widely used for univariate and multivariate forecasting tasks.

B.3.2 Large Pretrained and Fine-tuned Models

Chronos Ansari et al. (2024) introduces a large pretrained time series model leveraging transformer architectures pretrained on massive multivariate time series datasets across domains such as energy, finance, and healthcare. It typically has 100+ million parameters and demonstrates strong zero-shot and few-shot transfer learning capabilities.

TimeGPT TimeGPT (Garza et al., 2023) adapts the GPT architecture large transformer architecture for autoregressive time series forecasting. It is pretrained on vast temporal datasets, such as electricity usage and sensor data, and contains over 200 million parameters. This enables TimeGPT to generate high-quality forecasts and adapt through fine-tuning to various downstream tasks.

TimeFM TimeFM (Das et al., 2024) integrates factorization machines with transformer-based architectures to model higher-order interactions in temporal data efficiently. The pretrained model usually consists of around 50-100 million parameters, balancing expressiveness with computational efficiency.

Moments Moments Goswami et al. (2024) models temporal dynamics by learning statistical moments (e.g., mean, variance) of features in a pretrained setting. The model size varies but can reach 80 million parameters for deep architectures, allowing it to capture complex temporal dependencies.

B.4 Explainer Baseline Details

In this section, we provide additional details about the baseline explainers referenced in the main paper. Due to space limitations, some of these methods were only briefly mentioned. Here, we include a broader set of widely used explainers, along with information about their implementation in our codebase.

Gradient (GRAD). Baehrens et al. (2010) calculates the sensitivity of the output with respect to the input feature by taking the partial derivative.

Integrated Gradients (IG). Sundararajan et al. (2017) computes the path integral of gradients from a baseline input $\tilde{\mathbf{x}}$ to the actual input $\mathbf{x}$, scaling the difference between these inputs by the averaged gradient.

DynaMask. DynaMask (Crabbé & Van Der Schaar, 2021) is a perturbation-based explainer designed specifically for time series. It learns a continuous-valued mask to deform input signals towards a predefined baseline, using iterative occlusion to uncover the contribution of different time regions.

WinIT. WinIT (Leung et al., 2023) extends perturbation-based explainability to time series by focusing on the impact of feature removal across time steps. The explainer identifies time segments that, when masked, cause significant deviations in the model's prediction. A key component of WinIT is a generative model that reconstructs masked features to maintain in-distribution data during occlusion. This framework improves upon earlier explainers such as FIT, which we omit due to WinIT's stronger empirical and conceptual performance.

CoRTX. The CoRTX framework (Chuang et al., 2023) is a contrastive learning-based explainer originally developed for visual tasks. It approximates SHAP values by training an encoder through contrastive objectives involving perturbed versions of the input. For time series, we adapt CoRTX by applying it to temporal encoders and explainability modules. Although CoRTX and TimeX both utilize self-supervised learning, they differ in several respects. CoRTX relies on handcrafted augmentations and attempts to match SHAP scores, while TimeX and TimeX++ use masked binary classification (MBC) to guide mask learning without needing externally-generated explanations or fine-tuning. However, this may introduce out-of-distribution (OOD) samples for the predictor, which does not occur in the case of `TimeSAE`.

Saliency-Guided Training Ismail et al. (2021). SGT introduces interpretability directly into the training pipeline. The method iteratively masks out input regions with low gradient magnitudes, thereby encouraging the model to focus on salient features during learning. Although SGT modifies model training rather than offering explanations post hoc, we include it for completeness as it represents an in-hoc explainer. For evaluation, we apply gradient-based saliency maps as recommended

by the original authors. This method demonstrates that architectural or training modifications can promote more interpretable representations, offering an interesting point of comparison to TimeX and TimeX++.

CounTS. CounTS Yan & Wang (2023) is a self-interpretable time series prediction model that generates counterfactual explanations by modeling causal relationships among input, output, and confounding variables. It uses a variational Bayesian framework to produce actionable and feasible counterfactuals, providing causally valid and theoretically grounded explanations while maintaining predictive accuracy in safety-critical applications. We note that, our approach is related to CounTS but differs by functioning as a black-box explainer that does not require access to the internal model architecture or parameters, enabling broader applicability and flexible deployment across various time series models.

TimeX. TimeX (Queen et al., 2023) is an explainability method for time series models based on the information bottleneck (IB) principle. It extracts salient sub-sequences by optimizing a trade-off between informativeness and compactness. However, it suffers from out-of-distribution sub-instances and potential signaling issues, leading to explanations that may not be reliable or consistent.

TimeX++. Liu et al. (2024b) extends TimeX by introducing a modified IB objective that replaces mutual information terms with more practical and stable proxies. It generates explanation-embedded instances that are both label-consistent and within the original data distribution, significantly improving the fidelity and interpretability of explanations across diverse time series datasets.

Random attribution is used as a baseline control by assigning feature importance scores randomly for comparison.

B.5 TimeSAE Architecture Model

We use the following architectures for the Encoder and Decoder. Note that for different datasets we customize the latent dimension d, and we train with different latent dimensions.

Table 6: Encoder and Decoder Architectures of TimeSAE.

Layer	Size / Dimensions	Description
	Encoder	
TCN Stack	512-dim output	Time Convolution Network up to penultimate layer
Fully Connected Block (×5)	512 × 512	Linear layer + BatchNorm + Leaky ReLU (0.01) + Squeeze-and-Excitation (SE) block
Final Linear	512 × d	Outputs latent representation, followed by BatchNorm
	Decoder	
Linear	d → H	Initial fully connected layer (H = attention head dimension, e.g., 256 for ECG, 512 for ETTH1) + BatchNorm + Leaky ReLU
Fully Connected Block (×5)	H → H	Each block: Linear + Multi-Head Attention + BatchNorm + Leaky ReLU + SE block
Reshape	–	Reshape output into intermediate sequence for temporal reconstruction
Upsampling Stack	–	Upsampling layers (scale factor 2)
1D Convolutions	64 → 32 → C	Conv layers with decreasing feature maps; Leaky ReLU after each

B.6 Concept Interactions via TimeSAE 's Decoder.

We demonstrate the versatility of `TimeSAE` by applying it to a time series *forecasting* task. Specifically, we evaluate on the ETTh1 and ETTh2 datasets using both standard Transformer-based forecasting models and Large Pretrained Models. To adapt `TimeSAE` for this task, we first project the input time series $\mathbf{x} \in \mathbb{R}^{C \times T}$ into a latent concept space. For each forecast, we extract the

associated concept embeddings and use our decompositional decoder to reconstruct the input and attribute the forecast to specific concepts and time steps as demonstrated below Equation (13):

$$\boldsymbol{g}(\mathbf{c}) := \boldsymbol{\psi}_0 + \sum_{j=1}^{d} \boldsymbol{\psi}_1(\boldsymbol{c}_j) + \sum_{j=1}^{d-1} \boldsymbol{\psi}_2(\boldsymbol{c}_j, \boldsymbol{c}_{j+1}) + \sum_{j=1}^{d-2} \boldsymbol{\psi}_3(\boldsymbol{c}_j, \boldsymbol{c}_{j+1}, \boldsymbol{c}_{j+2}) + \cdots + \boldsymbol{\psi}_d(\mathbf{c}) \tag{13}$$

In the decomposition equation 13, the function $\boldsymbol{\psi}_k$ corresponds to interactions of order k, acting on k-tuples of input components. The index $k \in \{0, 1, \ldots, d\}$ denotes the order of interaction, where $k = 0$ corresponds to a constant term. For each fixed order k, the index $j \in \{1, \ldots, d-k+1\}$ refers to the position of the k-tuple within the input sequence $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$. Thus, $\boldsymbol{\psi}_k(\boldsymbol{c}_j, \ldots, \boldsymbol{c}_{j+k-1})$ operates on the contiguous subsequence starting at position j with length k. Specifically, for $k \in \{0, 1, \ldots, d\}$, $\boldsymbol{\psi}_k(\mathbf{c}_j, \ldots, \mathbf{c}_{j+k-1}) \in \mathbb{R}^{C \times T}$ denotes the contribution of the k-tuple starting at position $j \in \{1, \ldots, d-k+1\}$ in the input sequence $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_d)$. Each $\boldsymbol{\psi}_k$ is further factorized as an element-wise product

$$\boldsymbol{\psi}_k(\mathbf{c}_j, \ldots, \mathbf{c}_{j+k-1}) = \mathbf{h}_k(\mathbf{c}_j, \ldots, \mathbf{c}_{j+k-1}) \odot m_k^j, \tag{14}$$

where $\mathbf{h}_k(\mathbf{c}_j, \ldots, \mathbf{c}_{j+k-1}) \in \mathbb{R}^{C \times T}$ is a features computed from the k-tuple inputs, and $m_k^j \in \mathbb{R}^{C \times T}$ is a mask vector modulating $\mathbf{h}_k$ element-wise. Access to such a mask is particularly valuable, as it allows for direct comparison with other masking-based explanation methods. Unlike input-masking approaches, our mask is inherently robust to out-of-distribution (OOD) samples because it captures the concepts learned directly by the explainer. Moreover, our mask $\boldsymbol{m}_d$, which integrates all learned concepts, is analogous to those produced by methods such as DynaMask, TimeX, or TimeX++. An illustration is given in Figure 6 and Figure 5.

Example 1. In Figure 5 and Figure 6 we illustrate qualitative explanations for forecasting in ETTH1. A heatmap over the 48-hour historical context highlights the saliency of each input time step, indicating which regions contribute most to the forecast. To the right, the predicted values are shown over a fixed 24-hour forecast horizon.

We observe several consistent patterns. First, `TimeSAE` tends to identify the later input time steps as more influential, which aligns with the low temporal variability characteristic of the ETTH1 dataset. In the illustrated examples, the explanations provided by different models vary noticeably. For instance, in TimeSAE-TopK and TimeSAE-JumpReLU, the Informer model attributes influence to time steps around 10, 30, and the end of the context window. In contrast, large pre-trained models namely Chronos, TimeGPT, Moments, and Transformer tend to emphasize the final segments of the context. *This difference may be due to the learned temporal priors or positional encodings that emphasize recency and pattern consolidation near the prediction boundary.* These findings demonstrate that `TimeSAE` provides interpretable and time-localized explanations, shedding light on how latent concepts and temporal structures influence model predictions.

B.7 Algorithms

In Section 3, we introduced the alignment procedure. Here, we describe in detail how it can be performed, following the steps outlined in Algorithm 2.

B.7.1 Alignment

To support interpretability within our `TimeSAE` framework, we adopt a methodology inspired by Concept Activation Vectors (CAVs) (Lundberg & Lee, 2017) to align learned latent features (concepts) with human-interpretable notions. This alignment process involves associating dimensions in the latent space with meaningful, predefined concepts, thereby enabling post hoc explanation of model behavior. The alignment procedure consists of the following steps:

- **Concept Dataset Construction:** We first define a set of interpretable, low-level concepts. These can be manually annotated or derived using heuristics relevant to the domain. For each concept $\boldsymbol{c}_i$, we construct a labeled dataset composed of two sets of samples: those in which concept $\boldsymbol{c}_i$ is present (*positive set*) and a matched set of randomly selected samples where $\boldsymbol{c}_i$ is absent (*negative set*).

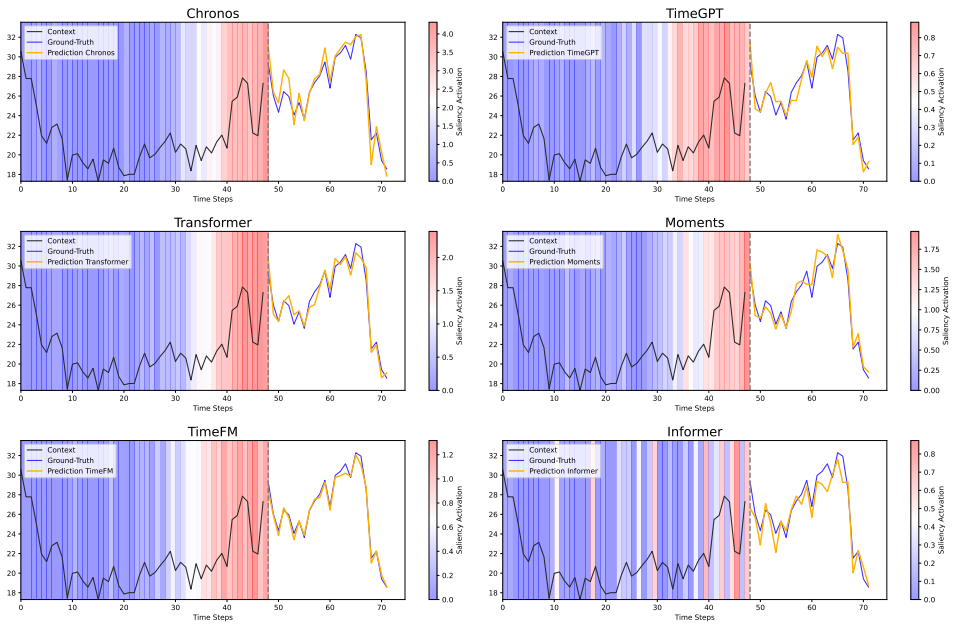

Figure 5: 24-hour forecast based on a 48-hour history from the ETTh1 dataset. The heatmap visualizes model explanations generated by TimeSAE-TopK for various forecasting models detailed in Section B.3.1.

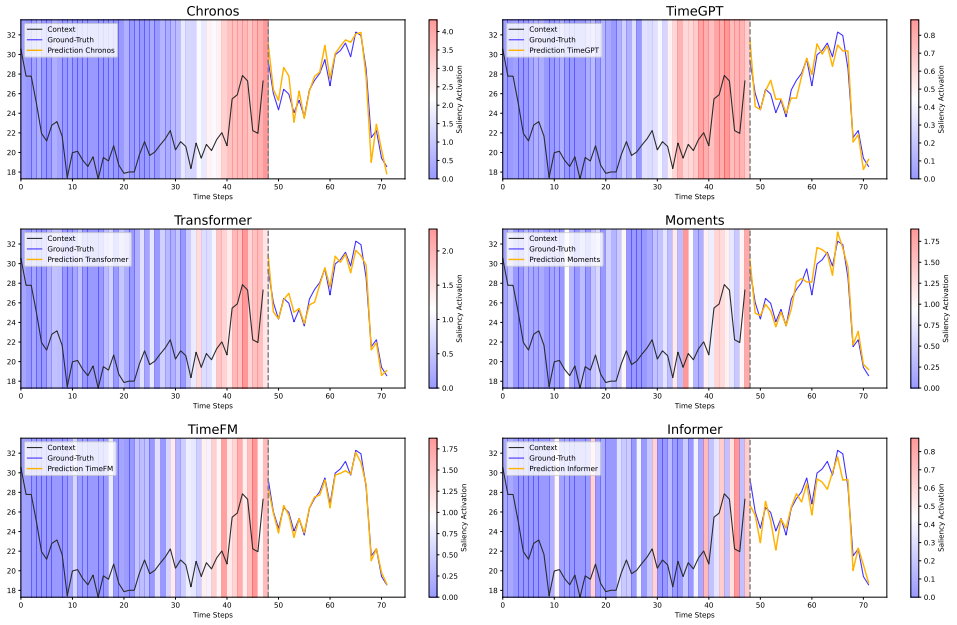

Figure 6: 24-hour forecast based on a 48-hour history from the ETTh1 dataset. The heatmap visualizes model explanations generated by TimeSAE-JumpReLU for various forecasting models detailed in Section B.3.1.

Algorithm 1 Generating Counterfactuals by Minimal Intervention on Selected Latent Concepts

Input: input $\mathbf{x}$, model $(\mathcal{E}, \boldsymbol{g})$, prediction $\mathbf{y}^{pred} = \boldsymbol{g}(\mathbf{x})$, target $\mathbf{y}^{cf} \neq \mathbf{y}^{pred}$, tolerance ϵ, learning rate w
Encode input to latent concepts via $\mathcal{E}$
Initialize intervention vector $\Delta\mathbf{c} = \mathbf{0}$
Select a subset of concepts $\mathcal{C}' \subseteq \{\boldsymbol{c}_k\}$ to intervene (e.g., those most influential)
while $|\boldsymbol{g}(\mathbf{c} + \Delta\mathbf{c}) - \mathbf{y}^{cf}| > \epsilon$ **do**
Compute gradients only for selected concepts:

$$\nabla_{\Delta \boldsymbol{c}_k} \mathcal{L} = \frac{\partial}{\partial \Delta \boldsymbol{c}_k} \left| f\big(\boldsymbol{g}(\mathbf{c} + \Delta\mathbf{c})\big) - \mathbf{y}^{cf} \right|, \quad \forall \boldsymbol{c}_k \in \mathcal{C}' \tag{15}$$

$$\Delta \boldsymbol{c}_k \leftarrow \Delta \boldsymbol{c}_k - w \cdot \nabla_{\Delta \boldsymbol{c}_k} \mathcal{L}, \quad \forall \boldsymbol{c}_k \in \mathcal{C}' \tag{16}$$

▷ Update interventions only on $\mathcal{C}$

Decode to get counterfactual:

$$\mathbf{x}^{cf} = \boldsymbol{g}(\mathbf{c} + \Delta\mathbf{c}) \tag{17}$$

return $\mathbf{x}^{cf}$

Algorithm 2 Concept Alignment using CAR and SVM

Require: Trained model M, dataset $\mathcal{D}$, concept set $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$, target layer L
Ensure: Concept-to-activation alignment models
for each concept $\boldsymbol{c}_i \in \mathcal{C}$ **do**
Define positive sample set $P_i \subset \mathcal{D}$ containing concept $\boldsymbol{c}_i$
Sample negative set $N_i \subset \mathcal{D}$ not containing $\boldsymbol{c}_i$
Initialize empty dataset $\mathcal{A}_i \leftarrow \emptyset$
for each $x \in P_i$ **do**
$a \leftarrow M_L(x)$ ▷ Extract activation from layer L
Append $(a, 1)$ to $\mathcal{A}_i$ ▷ Label 1 for positive
for each $x \in N_i$ **do**
$a \leftarrow M_L(x)$
Append $(a, 0)$ to $\mathcal{A}_i$ ▷ Label 0 for negative
Train SVC or SVR on $\mathcal{A}_i$ to distinguish presence of $\boldsymbol{c}_i$
Save model as alignment for concept $\boldsymbol{c}_i$
return Set of trained concept alignment models

- **Training Concept Classifiers:** Given the activations of the encoder for the above sample sets, we train a linear classifier (e.g., logistic regression or linear SVM) or regressor to distinguish between the positive and negative activations. The resulting weight vector defines a *Concept Activation Vector* (CAV), which serves as a direction in latent space that correlates with the presence of concept $\mathbf{c}$.
- **Computing Concept Scores:** For any test sample, we compute the similarity between its latent representation and the CAVs. This similarity (e.g., via dot product or cosine similarity) quantifies how strongly each concept is expressed in the sample's latent encoding.
- **Generating Explanations:** By projecting latent activations onto aligned CAVs, we can interpret which concepts are active for a given input. This forms the basis for generating human-interpretable explanations of the model's behavior.

This process enables a semi-automated way of auditing the latent space, identifying which learned dimensions correspond to known or meaningful concepts. Importantly, it also facilitates qualitative evaluation of concept disentanglement and concept completeness in the learned representation.

B.8 Hyperparameter Setting

We list hyperparameters for each experiment performed in this work. For the ground-truth attribution experiments (Section 4, for the synthetic dataset Figure 2 and the real-world dataset Table 9), the

Table 7: Training hyperparameters for TimeSAE-TopK across synthetic and real-world datasets used in our experiments for Transformer Predictor Yun et al. (2021).

Category	Dataset	r	Consistency weight α	Counterfactual weight λ	γ [min, max]	LR	Dropout	Batch size	Weight decay	Epochs
Synthetic	FreqShapes	1.5	0.8	0.9	[1, 10]	1e-3	0.1	64	0.01	100
	SeqComb-UV	1.7	1.0	0.8	[1, 8]	1e-3	0.25	128	0.001	300
	SeqComb-MV	1.6	0.9	1.0	[1, 12]	5e-4	0.25	128	0.001	300
	LowVar	1.4	0.8	0.9	[1, 9]	1e-3	0.25	64	0.001	150
Real-World	ECG	1.6	1.0	0.9	[1, 7]	2e-3	0.1	64	0.001	200
	ETTH1	1.5	0.9	0.8	[1, 11]	1e-4	0.1	64	0.001	300
	ETTH2	1.4	0.8	1.0	[1, 10]	1e-4	0.1	64	0.001	300
	PAM	1.7	0.9	0.9	[1, 9]	1e-3	0.25	128	0.01	100
	EliteLJ	1.5	1.0	0.8	[1, 12]	1e-3	0.25	64	0.001	500

Table 8: Training hyperparameters for TimeSAE-JumpReLU across synthetic and real-world datasets used in our experiments for Transformer Predictor (Yun et al., 2021).

Category	Dataset	r	Consistency weight α	Counterfactual weight λ	LR	Dropout	Batch size	Weight decay	Epochs
Synthetic	FreqShapes	1.6	0.85	0.9	1.2e-3	0.12	64	0.01	100
	SeqComb-UV	1.5	0.95	0.85	9e-4	0.3	128	0.001	300
	SeqComb-MV	1.7	0.9	1.0	6e-4	0.2	128	0.001	300
	LowVar	1.4	0.8	0.95	1.1e-3	0.22	64	0.001	150
Real-World	ECG	1.5	1.0	0.9	1.8e-3	0.15	64	0.001	200
	ETTH1	1.7	0.9	0.8	1.1e-4	0.11	64	0.001	300
	ETTH2	1.6	0.85	1.0	1.3e-4	0.13	64	0.001	300
	PAM	1.5	0.92	0.88	9.5e-4	0.28	128	0.01	100
	EliteLJ	1.6	1.0	0.82	1.0e-3	0.27	64	0.001	500

hyperparameters are listed in Table 7 and for TimeSAE-JumpReLU in Table 8. The hyperparameters used for the ablation experiment (Section 5.2, and Figure 4 with real-world datasets are in Table 7. We also list the architecture and hyperparameters for the predictors trained on each dataset in the Tables.

Selection of Dictionary Size r. The dictionary size r plays a key role in the performance of `TimeSAE` . To assess its impact, we perform an ablation sturdy on performance of the explanation of `TimeSAE` for both variates i.e. TopK and JumpReLU by varying r and evaluating the corresponding explanation quality across all datasets. Results are summarized in the Figure 7.

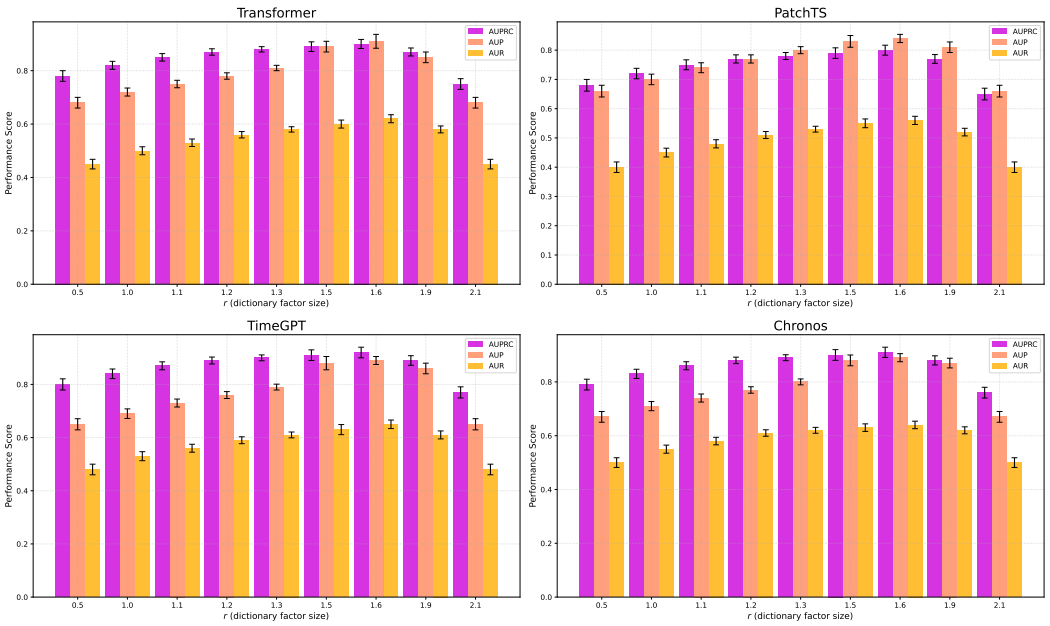

Figure 7: Impact of the hyperparameter r on model performance. Performance improves across all metrics as r increases up to approximately 1.6 for Transformer, 1.5 for PatchTS and TimeGPT, and 1.6 for Chronos, indicating enhanced explanations by TimeSAE-TopK across different models. Beyond values around 1.9, performance deteriorates, likely due to high sparsity. We note that higher metric values correspond to better performance.

TopK$_\gamma$ Scheduling. In Section 3, we introduced the use of the scheduler γ for training our variate TimeSAE-TopK. We now define its implementation. We also observe that the model fails to converge when using the originally proposed Multi-TopK (Gao et al., 2024) approach, which was intended to progressively cover concepts and mitigate saliency shrinking. In our case we define an integer scheduler $\gamma(t)$ at each training training step (out of a total number of steps) defined such that its value decreases from an initial integer $\gamma_{\max}$ down to 1, according to:

$$\gamma(t) = \max\left(1,\ \text{round}\left(\gamma_{\max} - \frac{t}{T} \times (\gamma_{\max} - 1)\right)\right) \tag{18}$$

where t is the current training step, $0 \leq t \leq T$, and T is the total number of training steps. The $\gamma_{\max}$ is the initial γ value ≥ 1 (e.g., 3). For each dataset, we specify the values of $\gamma_{\max}$ in Table 7.

B.9 Further Ablation Experiments

B.9.1 Ablation A1 - SAEs should be sparse, but not too sparse.

We investigate the effect of the latent dimension r, which determines the number of concept units in TimeSAE and indirectly influences explanation sparsity, as analyzed in our ablation study in the main paper. Sparsity aids in identifying concepts, as illustrated in Figure 10. As shown in Figure 7, increasing r from low values (e.g., 1.3) up to around 1.5-1.6 leads to consistent improvements across all evaluation metrics. This indicates that a moderately larger concept space allows the model to discover richer and more meaningful structures, resulting in better explanations. However, as r continues to increase (e.g., toward 2.0 or more), performance drops, likely due to reduced sparsity and the introduction of redundant or noisy concepts. Although r does not directly encode sparsity, it modulates how selectively the model can activate concept units. A smaller r

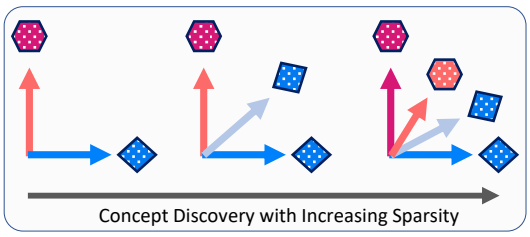

Figure 10: Intuition behind the effect of sparsity.

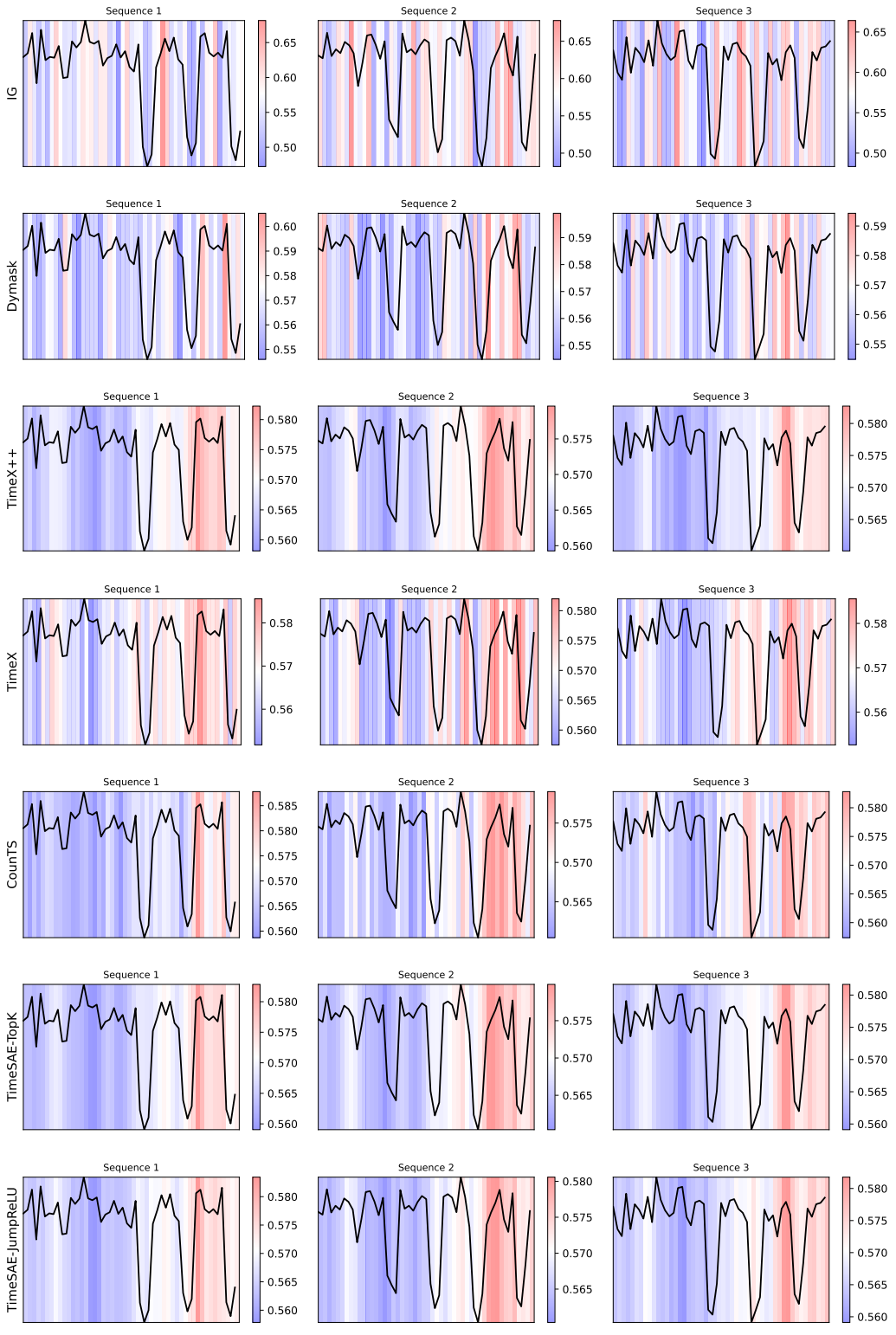

Figure 8: Visualization Explanation for the Transformer model's predictions on the FreqShapes dataset. From top to bottom: IG, DynaMask, TimeX, and TimeX++ illustrate saliency-based learning masks. CounTS represents counterfactual explanations. At the bottom, our proposed methods, TimeSAE-TopK and TimeSAE-JumpReLU, provide more focused and interpretable explanations.

Table 9: AUPRC explanation performance (higher is better) across methods for each dataset. For all metrics, higher values are better, and the colors represent the top **Top-1**, Top-2, and Top-3 rankings.

Black-Box	Dataset	IG	Dynamask	WinIT	CoRTX	TimeX	TimeX++	CounTS	TimeSAE
Transformer	ECG	0.788±0.041	0.310±0.066	0.505±0.022	0.707±0.018	0.533±0.021	0.925±0.037	0.916±0.027	**0.950±0.011**
	PAM	0.827±0.043	0.326±0.069	0.530±0.023	0.742±0.019	0.560±0.022	0.971±0.039	0.962±0.028	**0.998±0.012**
	ETTh-1	0.615±0.032	0.242±0.052	0.394±0.017	0.552±0.014	0.416±0.016	0.714±0.021	0.722±0.029	**0.741±0.009**
	ETTh-2	0.694±0.036	0.273±0.058	0.444±0.019	0.622±0.016	0.469±0.018	0.814±0.033	0.806±0.024	**0.836±0.010**
	EliteLJ	0.709±0.037	0.279±0.059	0.455±0.020	0.636±0.016	0.480±0.019	0.833±0.033	0.824±0.024	0.841±0.016
PatchTS	ECG	0.812±0.042	0.319±0.068	0.520±0.023	0.728±0.019	0.549±0.022	0.954±0.038	0.944±0.028	**0.980±0.011**
	PAM	0.852±0.044	0.336±0.071	0.546±0.024	0.765±0.020	0.870±0.040	0.902±0.023	0.882±0.029	**0.981±0.012**
	ETTh-1	0.634±0.033	0.249±0.054	0.406±0.018	0.569±0.014	0.428±0.017	0.734±0.022	0.744±0.030	**0.762±0.009**
	ETTh-2	0.715±0.037	0.281±0.060	0.458±0.020	0.641±0.017	0.483±0.019	0.830±0.025	0.839±0.034	**0.861±0.010**
	EliteLJ	0.731±0.038	0.288±0.061	0.469±0.021	0.700±0.017	0.494±0.020	0.859±0.034	0.850±0.025	0.866±0.027
TimeGPT (Pretrained)	ECG	0.756±0.073	0.298±0.118	0.485±0.039	0.679±0.032	0.512±0.037	0.883±0.066	0.892±0.048	**0.912±0.020**
	PAM	0.794±0.077	0.313±0.123	0.509±0.037	0.712±0.032	0.538±0.039	0.923±0.069	0.913±0.050	**0.957±0.022**
	ETTh-1	0.592±0.059	0.234±0.097	0.373±0.031	0.531±0.027	0.392±0.031	0.704±0.053	0.699±0.037	0.711±0.025
	ETTh-2	0.664±0.064	0.267±0.104	0.426±0.032	0.597±0.028	0.450±0.032	0.756±0.043	0.772±0.059	0.782±0.028
	EliteLJ	0.681±0.066	0.268±0.105	0.436±0.032	0.610±0.028	0.461±0.034	0.805±0.059	0.791±0.043	0.805±0.038
Chronos (Pretrained)	ECG	0.741±0.056	0.292±0.091	0.476±0.030	0.664±0.025	0.501±0.028	0.866±0.051	0.873±0.037	**0.894±0.015**
	PAM	0.779±0.059	0.307±0.095	0.499±0.036	0.698±0.025	0.527±0.030	0.905±0.053	0.887±0.038	**0.939±0.017**
	ETTh-1	0.580±0.045	0.229±0.075	0.365±0.024	0.520±0.021	0.384±0.024	0.689±0.041	0.678±0.028	**0.712±0.012**
	ETTh-2	0.651±0.049	0.262±0.080	0.417±0.025	0.586±0.022	0.441±0.025	0.749±0.045	0.733±0.033	**0.784±0.014**
	EliteLJ	0.667±0.051	0.263±0.081	0.427±0.025	0.598±0.022	0.452±0.026	0.767±0.033	0.788±0.045	**0.799±0.016**

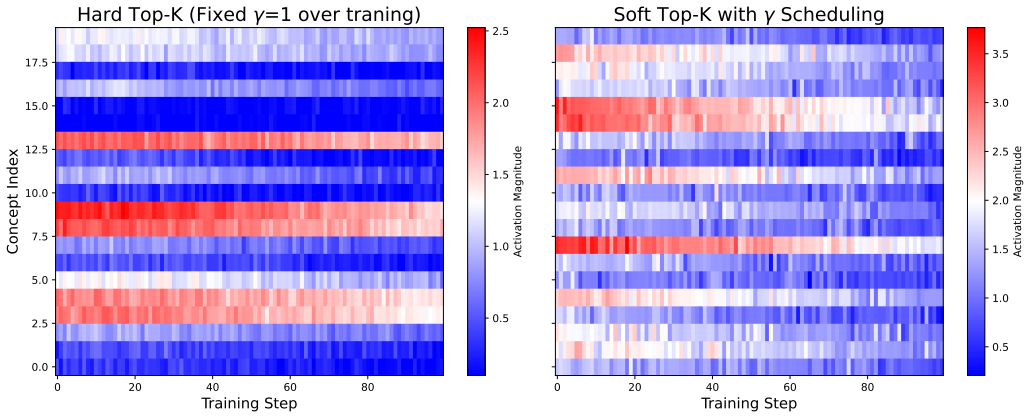

Figure 9: **Left:** Concept activations over training steps using hard Top-K with a fixed high γ value, evaluated on the same fixed validation time series sequence across training steps on the SeqComb-UV dataset, to explain the vanilla Transformer. As training progresses, many concepts exhibit near-zero activations, indicating the emergence of *"dead"* concepts that stop learning effectively. **Right:** Concept activations using soft Top-K with γ scheduling, where γ goes from 15 to 1 throughout training while having 20 concepts. This scheduling keeps activations dynamic and distributed across concepts, preventing *"dead"* concepts. The γ scheduling smooths the TopK selection, allowing gradual sparsification and enabling concepts to remain learnable, while a fixed γ imposes harsh sparsity that kills many latent features early.

naturally enforces stronger selection, while a larger r can dilute sparsity. These findings suggest that there is a sweet spot for concept dimensionality, where representations are expressive enough to explain predictions but sparse enough to remain interpretable.

B.9.2 Ablation A2 - TopK prevents activation shrinkage

This ablation study investigates how the use of TopK selection in the TimeSAE architecture mitigates the issue of activation shrinkage, which commonly leads to "dead" concepts that cease to learn during training. As shown in Figure 9, employing a fixed, high γ value with hard Top-K results in many concept activations collapsing to near zero over training steps, indicating that these concepts become inactive and contribute little to the model's interpretability or performance. In contrast, the soft Top-K variant with a scheduled γ that gradually decreases from 15 to 1 maintains more evenly distributed

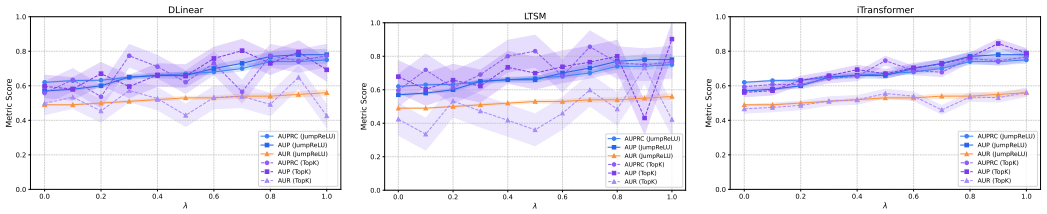

Figure 11: Extended ablation study on the effect of the concept consistency weight α for the EliteLJ dataset, evaluating metrics AUPRC, AUP, and AUR across TimeSAE models. Left: DLinear, most metrics perform well at $\alpha = 0.9$. Middle: LSTM, shows similar behavior to DLinear. Right: iTransformer, performance improves as α increases. Solid lines represent TimeSAE-TopK, dashed lines represent JumpReLU, and shaded areas indicate standard deviations over 10 runs. Slightly higher α values lead to more robust explanations.

and dynamic concept activations, thereby preventing concept "death." This gradual sparsification approach ensures that concepts remain responsive and learnable throughout training. Complementing this, Figure 8 visually demonstrates that the explanations generated by TimeSAE-TopK produce more focused and interpretable saliency maps compared to other black-box methods such as IG, DynaMask, and TimeX variants. These results collectively highlight that the TopK mechanism with γ scheduling not only preserves concept vitality by preventing activation shrinkage but also enhances the clarity and quality of explanations in Transformer-based time series models.

B.10 IMPLEMENTATIONS

B.10.1 PSEUDO CODE OF TIMESAE.

```
def timesae_jumprelu(params, x, sparsity_coefficient, use_pre_enc_bias):
    """
    Computes the forward pass and total loss for a JumpReLU-based Sparse Autoencoder.

    Args:
        params: Object containing model parameters (weights, biases, log threshold).
        x: Input batch (tensor).
        sparsity_coefficient: Scaling factor for the sparsity regularization term.
        use_pre_enc_bias: Boolean indicating whether to subtract decoder bias from input.

    Returns:
        Scalar representing the mean loss over the input batch.
    """
    if use_pre_enc_bias:
        x = x - params.b_dec

    pre_activations = relu(x @ params.W_enc + params.b_enc)

    # Compute threshold from the learnable log value
    threshold = exp(params.log_threshold)

    # Apply JumpReLU for sparsity-aware feature extraction
    feature_magnitudes = jumprelu(pre_activations, threshold)
    # decoding
    x_reconstructed = feature_magnitudes @ params.W_dec + params.b_dec

    # Compute reconstruction loss
    reconstruction_error = x - x_reconstructed
    reconstruction_loss = sum(reconstruction_error ** 2, axis=-1)

    # Compute L0-style sparsity penalty
    l0_penalty = sum(step(feature_magnitudes, threshold), axis=-1)
    sparsity_loss = sparsity_coefficient * l0_penalty

    total_loss = mean(reconstruction_loss + sparsity_loss, axis=0)
    return total_loss
```

B.10.2 PSEUDO CODE OF TIMESAE (CASE WITH TOPK)

```

def gamma_scheduler(step, max_gamma=10.0, min_gamma=1.0, total_steps=10000):
    """
    Exponential decay scheduler for $\gamma$.
    """
    progress = min(step / total_steps, 1.0)
    return max_gamma * (min_gamma / max_gamma) ** progress

def timesae_soft_topk(x, params, k, gamma, sparsity_coeff, use_pre_enc_bias):
    """
    Full loss computation for TimeSAE-TopK_gamma.
    Combines encoding, decoding, and loss into one compact function.

    Args:
        x: input batch [batch, features]
        params: model parameters (W_enc, b_enc, W_dec, b_dec)
        k: top-k features to retain
        $\gamma$: current $\gamma$ value (>1 early in training, → 1)
        sparsity_coeff: weight for L0 sparsity loss
        use_pre_enc_bias: if True, subtract b_dec before encoding

    Returns:
        Mean total loss (MSE + sparsity)
```

```
    """
    # Optional pre-encoder bias
    if use_pre_enc_bias:
        x = x - params.b_dec

    # Encode with ReLU
    pre_activations = relu(x @ params.W_enc + params.b_enc)

    # Compute $\gamma$-scaled TopK mask
    sorted_vals, _ = torch.sort(pre_activations, dim=-1, descending=True)
    threshold = sorted_vals[:, k - 1:k]  # shape [batch, 1]
    topk_mask = (pre_activations >= threshold).float()
    soft_mask = torch.clamp(gamma * topk_mask, max=1.0)

    # Apply sparsity
    sparse_features = pre_activations * soft_mask

    # Decode
    x_reconstructed = sparse_features @ params.W_dec + params.b_dec

    # Compute losses
    reconstruction_loss = ((x - x_reconstructed) ** 2).sum(dim=-1)
    sparsity_loss = sparsity_coeff * (sparse_features > 0).float().sum(dim=-1)

    return (reconstruction_loss + sparsity_loss).mean()
```

C LIMITATIONS

While TimeSAE provides faithful and interpretable explanations, several practical aspects offer exciting avenues for future research for the time series community:

[1] **Dependence on the Dataset Used to Train the Black-Box Models:** Our method benefits from access to sufficiently large datasets to effectively explain black-box models. In scenarios where data are limited, exploring strategies such as domain adaptation or leveraging similar but different distributions could enable TimeSAE to generalize well with fewer data. Developing such approaches can broaden applicability to data-scarce or specialized domains, opening up valuable research directions. We believe that relaxing certain assumptions can be a significant step toward developing more general explainable methods for time series. Moreover, the proposed approach offers a new perspective by leveraging Sparse Autoencoders (SAEs) as an explainability tool for time series, similar to their successful use in large language models for discovering highly interpretable concepts (Cunningham et al., 2023).

[2] **Sensitivity to Hyperparameter Settings:** Although hyperparameter choices like sparsity levels and dictionary size significantly impact performance, this also presents an opportunity to develop more automated, data-driven tuning methods. Advances in hyperparameter optimization or self-regularizing architectures could reduce manual effort and improve TimeSAE 's ease of use and transferability to new tasks.

D REPRODUCIBILITY

To ensure reproducibility of our experiments, we provide the complete source code, including data preprocessing scripts, model training routines, and evaluation metrics, at the following GitHub repository: https://anonymous.4open.science/w/TimeSAE-571D/. All experiments were conducted using fixed random seeds, on three A100 Nvidia GPUs. Detailed instructions for installing dependencies, configuring hyperparameters, and replicating the results presented in this paper are included in the repository's README file. Additionally, preprocessed datasets and pre-trained model checkpoints are provided to facilitate direct reproduction of the reported performance metrics.