Universiteit Leiden
The Netherlands

Data Science & Artificial Intelligence

Adaptation of Mechanistic Interpretability Methods to Time Series Transformers in Classification Tasks

Matiss Kalnare

Supervisors:
Dr. Niki van Stein & Kitharidis Sofoklis

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/07/2025

Abstract

Transformer models have shown strong performance on Time Series Classification (TSC) tasks, yet remain difficult to interpret. While most explainability methods focus on input-output attribution, they offer limited insight into the internal mechanisms that drive predictions. This thesis investigates whether mechanistic interpretability (MI) techniques, originally developed for NLP models, can be adequately adapted to transformer-based time series models. We apply a suite of MI methods to a Time Series Transformer (TST) model trained on the JapaneseVowels dataset, including activation patching, attention saliency, and sparse autoencoders. These techniques enable us to probe the causal roles of individual attention heads, specific timesteps, and learned latent features in the model's decision-making. By constructing causal graphs from patching results, we trace how information flows from input to output, revealing interpretable internal circuits. Additionally, we show that sparse codes learned from internal activations are not only class-discriminative but could themselves be patched, suggesting a path toward manipulating high-level concepts within the model. While not aiming for full interpretability, it demonstrates the technical viability and promise of applying MI to TSC, and lays the groundwork for more scalable and generalizable interpretability pipelines in future work.

Contents

1 Introduction

Deep learning has become a cornerstone for efficient and reliable solutions in many fields, ranging from computer vision to natural language processing [BCE+23], and more recently, in time series classification (TSC). Various models, such as convolutional neural networks and transformers, offer good performance, but often work as *black boxes*, giving no insight into their decision-making. The lack of ability to look inside the models raises valid concerns about trust, accountability, and safety [JQC+25]. This is becoming increasingly more of a problem as more and more models are deployed in real-world settings like healthcare every day.

The field of Explainable Artificial Intelligence (XAI) is focused on addressing these concerns by making model outputs more interpretable and understandable to humans. While XAI has made great progress in image and text domains, time series data remains comparatively unexplored despite its central role in many applications [GYS24, RPF+21].

Moreover, many current XAI techniques, namely SHAP and LIME, are model-agnostic, meaning they can explain predictions regardless of the model's internal structure. However, these explanations are only surface-level [CES+24]; they only highlight which features are important, not **why** or **how** a model reaches a decision. Therefore, a shift to internal mechanics of deep neural networks is key for inner interpretability [RHCHM23].

Mechanistic Interpretability is an approach that tries to gain insights into the internal workings of deep learning models by reverse engineering the network's inner computations at an increasing level of granularity. It has recently gained momentum in the study of transformer-based language models [ENO+21, FSBCj24, TCM+24].

1.1 Challenge and Goal

Researchers have made advancements in analyzing transformers by identifying the roles of specific layers, attention heads, and neurons, revealing how they relate to tasks such as factual recall [MBAB23]. However, this type of analysis is extremely time-consuming, requiring extensive manual effort, careful experimentation, and technical expertise [BG24].

At the same time, transformer-based models are being increasingly applied to Time Series Classification (TSC) due to their ability to capture long-range dependencies and contextual patterns [WZZ+23]. However, unlike natural language tasks, time series data does not have a clear token-based structure or semantic meaning, which makes reasoning about how the information flows through the model or what specific components are responsible for key decisions difficult.

As a result, this leads to the central point of this thesis: **Can mechanistic interpretability techniques used in NLP be adapted to transformer-based models operating on time series data?** Specifically, the research focuses on investigating and evaluating how the methods transfer to Time Series Transformer (TST) models on the TSC task. This thesis will employ certain techniques that intervene in a TST model's computation and observe how specific components contribute to its predictions.

Importantly, the objective is not to provide a full interpretation of a trained model or to draw definitive conclusions about its internal logic. Rather, the emphasis is on exploring whether these interpretability methods can be meaningfully adapted to the time series domain, and to what extent they offer insights. By testing technical viability, it lays the groundwork for future work that may use such methods for actual model analysis and domain-specific explanations.

1.2 Thesis Overview

This thesis is structured to guide the reader through the motivation, theoretical background, methodological adaptation, and evaluation of mechanistic interpretability techniques in the context of time series transformers.

Section 2 introduces key terminology and formal definitions that will be used throughout the thesis; Section 3 provides an overview of related work, covering developments in time series classification, transformer architectures, and research in explainable AI and mechanistic interpretability; Section 4 describes the setup, including the dataset used, the model architecture, and its training; Section 5 details the interpretability techniques adapted from NLP; Section 7 presents the results of applying the methods to a trained TST model and evaluates the extent to which the adapted techniques produce a meaningful insight; Section 8 offers a discussion of the findings, limitations, and broader implications for future research; Lastly, Section 9 concludes the thesis.

This thesis is part of the Bachelor of Data Science and Artificial Intelligence program at Leiden Institute of Advanced Computer Science (LIACS), Leiden University, under the supervision of Niki van Stein (LIACS) and Kitharidis Sofoklis (LIACS).

2 Definitions

Explainable Artificial Intelligence (XAI) A field of AI that aims to make the behavior and decisions of machine learning models more interpretable and understandable to humans.

Mechanistic Interpretability (MI) An approach within XAI focused on reverse-engineering the internal components of a neural network to understand how they lead to specific outputs. MI treats components such as layers, neurons, or attention heads as causal units and explores their roles using observational and interventional tools.

Transformer A neural network architecture based on self-attention mechanisms that enables modeling long-range dependencies. Initially developed for NLP, it has been adapted for various domains, including time series.

Time Series Classification (TSC) A supervised learning task where each input is a sequence of data points ordered in time, and the goal is to predict a class label for the entire sequence.

Time Series Transformer (TST) A specialized form of the Transformer architecture adapted for TSC tasks. It captures temporal patterns and interdependencies in multivariate time series data.

Activation Patching An interventional MI technique where activations from one example are copied into another example at specific locations in the model to observe how output changes, thereby revealing causal roles of components.

Attention Saliency An observational MI technique that analyzes the attention weights in a Transformer model to identify which input elements a specific attention head focuses on. It highlights potential areas of influence but does not establish causality.

Sparse Autoencoder (SAE) A neural network trained to reconstruct its input under a sparsity constraint on the latent space. In MI, SAEs are used to disentangle internal representations and potentially identify semantically meaningful features.

Causal Graph A directed graph that represents the flow of causal influence within a model, often built from patching results to visualize which components contribute to specific predictions.

Clean Instance An input instance for which the model makes a correct and confident prediction.

Corrupt Instance An input instance for which the model makes an incorrect or low-confidence prediction.

Denoising vs. Noising Two patching strategies: denoising involves patching clean activations into corrupt examples to test sufficiency, while noising involves patching corrupt activations into clean examples to test necessity.

3 Background & Related Work

To frame our contribution, this section surveys the evolution of TSC, the adaptation of Transformer architectures to temporal data, state-of-the-art explainability and mechanistic interpretability techniques, and their parallels in neuroscience. By covering these topics, we show the main challenges that drive our effort to adapt interpretability techniques for TSTs.

3.1 Time Series Classification

TSC refers to the task of labeling an entire temporal sequence. A time series can be univariate (only one variable, e.g., stock price of Apple (AAPL) over days) or multivariate (multiple variables, e.g., accelerometer readings (X, Y, Z axes) per second) and is structured as a sequence of observations sampled at equally spaced intervals. TSC plays a crucial role in various domains, some of the more notable ones are healthcare (e.g., ECG signal analysis) and finance (e.g., stock trend prediction)

Historically, non-deep learning methods like *k-Nearest Neighbors* and *feature-based classifiers* were most prominent in the field. Methods such as Catch22 [LSK+19] and ROCKET [DPW20] match or even outperform deep architectures such as ResNet [ZJP+21] at a fraction of computational cost. However, they often depend on handcrafted features, as well as the availability of **labeled data**, which is scarce, as labeling data often requires much time, expertise, and infrastructure.

Conversely, unlabeled data is abundant, prompting significant demand for methods that can deliver high accuracy with limited labeled data or by effectively utilizing the available unlabeled data [ZJP+21].

3.2 Transformers for Time Series

Originally proposed for natural language processing (NLP) tasks, transformers have rapidly found applications across diverse domains due to their superior capability in modeling long-range dependencies via self-attention mechanisms [WZZ+23]. This capability is particularly beneficial for time series, where relevant patterns may appear across non-adjacent time points, requiring effective modeling of temporal dependencies.

Recent literature highlights several advantages of applying transformers to time series classification and forecasting tasks, such as robustness in capturing complex temporal patterns and parallel computation benefits [ZJP+21, WZZ+23]. Additionally, transformers have demonstrated notable performance improvements over traditional architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [ZJP+21], indicating their potential as a leading approach in TSC.

However, applying transformers to time series requires adaptations to bridge the gap between language data and temporal signals. Due to this, unsurprisingly, transformers in time series also trail behind their NLP counterparts in terms of interpretability, as adaptations have to be made there too.

3.3 Explainable AI and Mechanistic Interpretability

Traditional explainability in deep learning has focused on attribution methods like SHAP [LL17] and LIME [RSG16], which assign importance scores to input features. These approaches work

on the *input-output* level, offering a limited view into the internal structure of deep networks. In time series, such methods fail to capture the temporal interactions and reasoning embedded in transformer models.

Mechanistic Interpretability (MI) aims to understand deep neural networks from the "*inside-out*". This paradigm sees the model's computation graph as a causal graph, where individual layers, neurons, or attention heads are treated as potential causal units, whose roles can be probed and analyzed. MI techniques can broadly be put into two categories [BG24].

First, **observational methods** analyze the internal behavior of a model without altering its computational flow. They can reveal statistical associations; however, they do not imply causation. *Attention Saliency* is one of the most used techniques in transformers. It inspects self-attention weights to infer which input positions or features the model focuses on [CKLM19]. However, saliency does not necessarily indicate causal influence as attention weights may not correlate with importance [SS19]. *Sparse Autoencoders* are used to extract interpretable latent features from hidden activations. They compress and reconstruct internal representations, allowing for disentangling and interpreting subspaces within a network [CER+23].

Second, **interventional methods** actively modify the model's internal computations to test hypotheses about causal mechanisms. One such technique is *activation patching*, which involves copying activations from a "*clean*" example and injecting them into a "*corrupt*" example at specific points in the model. This patch can cause the model's output to change and suggest causal influence on the original decision. This technique expands into *causal tracing* where by formalizing interventions and comparing runs with and without specific patches, one can distinguish between components that are *necessary* (i.e., patching corrupt $\rightarrow$ clean corrects behavior (AND logic) and *sufficient* (i.e., patching clean $\rightarrow$ corrupt introduces error (OR logic) for a specific computation [HN24]. Furthermore, this can be extended into *path tracing*, e.g., tracing how a signal flows from a specific input through multiple layers, allowing one to identify which paths are critical for a specific behavior [WVC+22].

Recently, MI has proven successful in revealing circuits for factual recall and other computations in large language models like GPT-2 [MBAB23, WVC+22], and even large-scale attribution-graph analyses of models such as Claude 3.5 [LGA+25]. Meanwhile, in the vision domain, mechanistic approaches are emerging to open the black box of Vision Transformers (ViTs). Fine-tuning ViTs on distorted images reveals that robustness can emerge through causal changes in attention heads and MLPs, with specific components becoming necessary to correct corrupted inputs [Bah25]. Additionally, the Prisma framework [JSH+25] introduces a unified open-source toolkit for probing internal ViT circuits via activation patching, sparse autoencoders, and neuron-level interventions, scaling MI workflows.

3.4 Mechanistic Interpretability Gaps in Time Series

A range of studies have attempted to bring interpretability to TSC models [IVS+24], but there are limited studies on transformer-based models. Moreover, most rely on black-box attribution methods, which produce input feature importance scores but do not reveal how internal components of the model contribute to the final prediction.

A potential reason for this gap can be attributed to fundamental differences in data structure. For one, time series lack tokenized semantic meaning, making it harder to create clean/corrupt contrast pairs. Furthermore, temporal locality is different from syntactic structure in text, com-

plicating reasoning about causal dependencies between attention heads or MLP layers. Another factor could be that many time series datasets are small and have limited scope for pertaining and large-scale interpretability studies.

While the field of MI encompasses a wide range of methods, this thesis examines the adaptation of activation patching and attention saliency to TST, as these methods have proven to be successful. Sparse Autoencoders will be briefly introduced and discussed as an auxiliary method to the above-mentioned methods. Nevertheless, the primary goal is to asses whether these techniques can provide meaningful insights into how TST models operate and possibly enrich the interpretability toolkit for time series.

3.5 Connection to Neuroscience

The *reverse engineering* approach of MI draws inspiration from a multitude of interdisciplinary fields, such as physics and systems biology, as well as neuroscience [BG24]. Moreover, the shift from previous interpretability methods to mechanistic interpretability can be compared to the change from behaviorism to cognitive neuroscience in psychology [BG24]. Interestingly, for our chosen MI methods, it is possible to draw some parallels from neuroscience. For example, transcranial magnetic stimulation (TMS) is a brain stimulation technique used in clinical settings to modulate neural activity. It uses magnetic pulses to induce an electric current in specific parts of the brain. These pulses can either excite or inhibit neural activity, or simply put, cause or disable neuron activations.

By inactivating or exciting a specific targeted brain area, scientists can observe behavioral effects and create causal mappings, establishing causal roles of brain regions in cognition (e.g., which part is involved in language, attention, motor function, etc.) [RRAG11][HGTH23]. TMS has also been shown to help patients with depression [TPT+25] and migraines [LP10] among other diseases. Activation patching in transformers can intervene in a similar sense to map parts of the transformer network to certain effects, as well as give an opportunity to fix the model's errors, akin to alleviating a disease in a patient.

Another comparison can be made with electroencephalography (EEG), which records the electrical activity of the brain and can also be used for brain mapping and therapeutic contexts [MWG+24]. As an example, EEG has been used by researchers to determine a normative brain map, which they then used to identify abnormal activity in epileptic patients [JOC+23]. In a similar sense, we can see the internal workings of a transformer in normal and abnormal situations using various observational MI techniques.

Lastly, *attention saliency* refers to understanding which part of an input a transformer model is attending to. In our case, it would be specific timesteps. Here we can look at magnetoencephalography (MEG), which tracks brain activity in very high temporal resolution (milliseconds) by measuring magnetic fields from active neurons [Gro19]. For example, researchers used MEG to track a listener's attentional focus in real-time in the *cocktail party problem* [APSB15]. And just like with interpretability in AI, the goal of the scientist is to decode specific information from brain signals instead of simply describing them [Gro19].

4 Setup

This section outlines the architecture of our transformer-based classifier, the characteristics of the JapaneseVowels dataset, and our training procedure. All code implementation for this setup (along with pretrained models), methodology, experiments, as well as results are publicly available on GitHub [Kal25].

4.1 Model Architecture

Our classifier f_θ processes each input

$$X \in \mathbb{R}^{T \times C},$$

where T is the length of the time series and C is the number of channels. Our model first applies three 1D-convolutions of shape

$$C \to \frac{d}{4}, \qquad \frac{d}{4} \to \frac{d}{2}, \qquad \frac{d}{2} \to d$$

and kernel size 5, 3, 3 respectively, where d is the model's hidden dimension, each followed by batch normalization and ReLU, to extract local features. Then we add positional embeddings

$$P \in \mathbb{R}^{T \times d},$$

to the convolved output and pass the resulting sequence into $L = 3$ transformer encoder layers. Each layer uses multi-head self-attention with $H = 8$ heads, followed by a two-layer feed-forward block, residual connections, dropout, and layer-normalization in standard transformer fashion [WZZ+23]. After the final encoder, we max-pool over the time axis to get a single $d-$vector, and feed that through a linear classification layer $\mathbb{R}^d \to \mathbb{R}^K$ followed by softmax to produce our K class probabilities.

The transformer's decoder portion of the original architecture is omitted. Decoders' typical job is to generate, whether it be the next word in a sequence in NLP, or forecasting in time series. However, it is unsuitable for classification, which has a fixed-length, non-autoregressive output [ZJP+21]. This comes with an added benefit of only using about half of the model parameters, reducing computational costs.

4.2 Dataset

To analyze the feasibility of MI techniques on TST, we need a suitable dataset to work with. The choice was made to work with **JapaneseVowels** [KTS99], summarized in Table 1 from UAE Time Series Classification Archive [BDL+18]. In this dataset, each instance is a 12-dimensional timeseries obtained by applying a 12th-order linear prediction analysis to raw recordings of nine male speakers each uttering the Japanese vowels "a" and "e", visualized in Figure 1. The task is to classify the identity of the speaker (labelled 1, ..., 9). The training set contains 30 utterances per person; the test set ranges from 24 to 88 utterances per person, with each utterance length varying between 7 and 29 frames.[1]

[1]The version provided by the `AEON` package, which we utilized in our code, standardizes all sequences to a fixed length of 25 time steps through truncation or padding to ensure uniform input shape.

Dataset	Train	Test	Series Length	# Classes	Dimensionality	Type
JapaneseVowels	270	370	25	9	12	Audio

Table 1: Summary of the JapaneseVowels dataset.

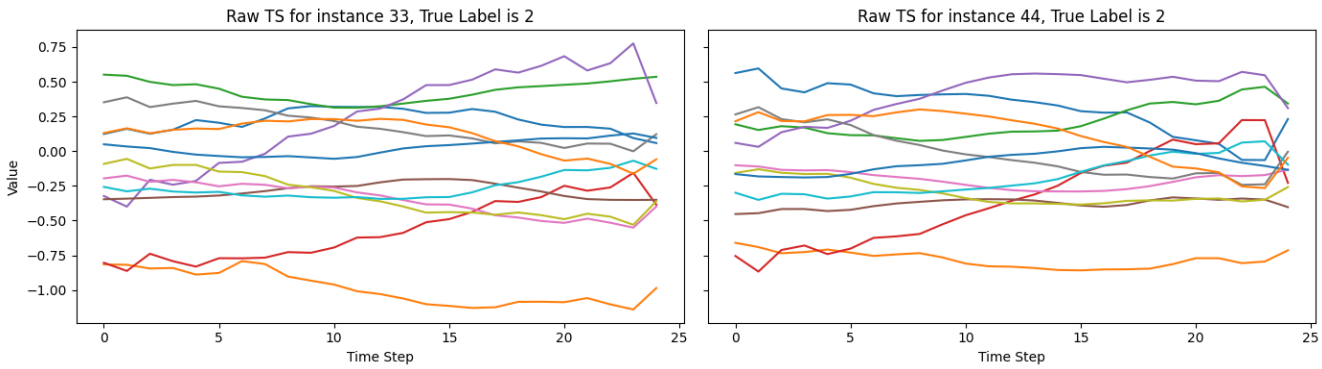


Figure 1: Sample visualizations of multivariate time series instances from the JapanseVowels dataset, showing 12-channel signal over 25 time steps.

The choice of this dataset is substantiated by the fact that it is multivariate, yet *low*-dimensional, allowing TST to exercise its ability to learn inter-channel dependencies via self-attention while still being able to follow along by humans. Furthermore, its short sequences enable rapid processing, which in turn allows for efficient testing of downstream interpretability techniques. Lastly, prior work highlighted high accuracy despite the small training size [ZJP+21], making it an ideal benchmark to highlight TST's strengths and investigate its inner workings.

4.3 Training Setup

Our defined TST model is trained on the JapaneseVowels dataset. Each time-series sample $X \in \mathbb{R}^{25\times 12}$ and its one-hot label $y \in \{1, \ldots, 9\}$ are batched in batch sizes of 4. The classifier f_θ is trained over 100 epochs with cross-entropy loss. For optimization, the Rectified Adam (RAdam) optimizer with a learning rate of 10^{-3} and weight decay of 10^{-4} is used.

This setup, with small batch size, intensive weight decay, and RAdam, gives stable convergence on the low-sample, multivariate dataset and is easily reproducible.

5 Methodology

The methodological pipeline employed in this thesis mirrors that of identifying and editing factual associations in transformer-based NLP models [MBAB23]. A clear, generalized version involves selectively replacing internal activations from a source instance, which correctly performs a specific task, into a target instance, which does not. Or the other way around. This action is referred to as **Activation Patching** (or simply *patching*).

The instance selection plays a large role and is very important in the case of NLP, where an instance is a specific prompt. Additionally, it is worth experimenting across different levels of granularity. That is, patching can be done for a certain layer, or a certain component of a layer, e.g., attention head, or even at the level of individual neurons. Moreover, there are different techniques, but the two main ones are *Denoising* and *Noising*. In denoising, activations are captured from a clean source and patched into a corrupt target. This observes which activations are **sufficient** to restore intended behavior. Noising, on the other hand, captures activations from a corrupt source and patches them into a clean target. In this case, we observe which activations are **necessary** to maintain the intended behavior. It is crucial to point out that the two are **not** symmetric mirrors of each other, and each approach has its use in different circumstances [HN24].

These core concepts of activation patching will now be adapted to the time series domain, specifically to our previously defined TST model for a classification task.

5.1 Instance Selection & Definition of Evaluation Metric

To effectively apply activation patching to a TST model, the first important step is to select source and target instances. We define clean instances as the instances from the test set, where the model predicts the correct class of that instance with a very high confidence (above 95%). Corrupt instances are the instances from the test set where the model misclassifies (the confidence for the correct class is below 50%). This selection of instances ensures a contrast between them, while maintaining meaningful similarities. Specifically, both clean and corrupt instances originate from the same class, ensuring their differences primarily reflect model recognition rather than inherent class-level variance.

For our primary metric of evaluating the impact of a patch, we choose the change in class prediction probability - ΔP. This is computed by first obtaining the baseline probability $P_i(y_{true})$ where i is the target instance and y_{true} is the true class label, through a standard forward pass of the model without any interventions, extracting logits, and subsequently applying a softmax to determine class probabilities. Then, after applying a patch, we compute the new portability $P_{i_{patched}}(y_{true})$. Thus, the full metric is expressed as:

$$\Delta P = P_{target\_patched}(y_{true}) - P_{target}(y_{true})$$

A positive ΔP shows an improvement in classification confidence, which can suggest a causal contribution of the patched component to correct classification.

5.2 Activation Patching Pipeline

The technique involves a procedure facilitated by `PyTorch` forward hooks, and we focus on the *Denoising* approach. The clean instance undergoes a forward pass, during which hooks are registered

on targeted model components. These hooks capture and cache activations at precise internal model locations, providing the clean baseline activations for subsequent patching. For the corrupt instances' forward pass, the cached activations from the clean instance are selectively injected into the corresponding model components, and the patched output logits are recalculated:

$$logits_{patched} = f_\theta(\text{corrupt instance} — \text{clean activations})$$

Followed by the new probability computation for the true class:

$$P_{patched}(y_{true}) = softmax(logits_{patched})[y_{true}]$$

Then the causal effect of the patching can be quantified by the above-defined metric of ΔP.

These patches were targeted at various granularity levels, namely, individual whole layers, followed by specific individual attention heads, finished off by position-specific (time steps) heads activations. The results of the patches can be of **AND**, where multiple components need to be patched simultaneously, or **OR** logic, where individual internal activation patches can sufficiently restore accuracy. A similar targeting can be done to MLP layers, i.e., the intermediate feed-forward layers.

Furthermore, we define a critical patch as a patch in the lowest granularity level - particular head, layer, and position - where the resulting change ΔP exceeds a set threshold in favor of the correct class. To systematically identify critical patches, we perform an exhaustive sweep across all combinations of layers, attention heads, and positions. For each configuration, the above-defined pipeline is followed, and ΔP is stored. All patches exceeding the set threshold are considered to be part of the critical patches set. This set is then used to build *causal* graphs to highlight specific attention head influence on the model output probabilities. These results, causal graphs, and other visualizations are provided in Section 7.

5.3 Attention Saliency

To accompany the activation patching, we employ *attention saliency* as an observational tool to clarify which timesteps of the instance the model is attending to with each attention head. For this, the self-attention layers are intercepted during a forward pass, and the raw attention weights $A \in \mathbb{R}^{H \times T \times T}$, where H is the number of attention heads and T is the sequence length, are captured.

To compute the timestep-level saliency for a given head, we extract $A^{(h)} \in \mathbb{R}^{T \times T}$ and average over the query axis:

$$S_t^{(h)} = \frac{1}{T} \sum_{i=1}^{T} A_{i,t}^{(h)}$$

This gives us the salience score $S_t^{(h)}$ for each timestep t, representing how much (on average) the model attends to that timestep across all queries in head h. Here we note that attention saliency does not imply causality, yet it is useful for highlighting timesteps and guiding hypotheses for causal experiments.

6 Sparse Autoencoders

As our patching experiments progress from full layers to individual attention heads and timesteps, a natural next step would be to investigate individual neurons in our TST encoder blocks. However, attempting to patch neurons in this layer directly poses several practical and conceptual challenges. Along with self-attention, each transformer encoder block consists of MLP blocks, which consist of two linear transformations with an activation and dropout layer in between. Each MLP block outputs a high-dimensional activation, producing thousands of neuron activations to potentially patch.

First, patching every neuron activation across instances is prohibitively expensive. For a single instance pair and encoder block in our TST, one must sweep across 3,000 positions. Second, and more importantly, neurons in these MLP layers are entangled [BG24]. Each unit is participating in a multitude of overlapping computations without a clear semantic boundary. There is no inherent ordering, modularity, or interpretability in the raw activation space (see Figure 2).

Sparse Autoencoders (SAEs) are introduced as an auxiliary method for analyzing and understanding internal representations in TST models. SAEs are neural networks trained on the internal activations of a specific layer within a transformer model. They are trained to reconstruct their inputs with enforced sparsity constraints on their hidden layer. This sparsity promotes the models' ability to discover a compact set of latent features, with the ultimate goal being the disentanglement of neurons, which can then reveal meaningful concepts in the data. These learned concepts can then be interpreted, visualized, and even patched into the forward passes of the main classifier to their causal influence.

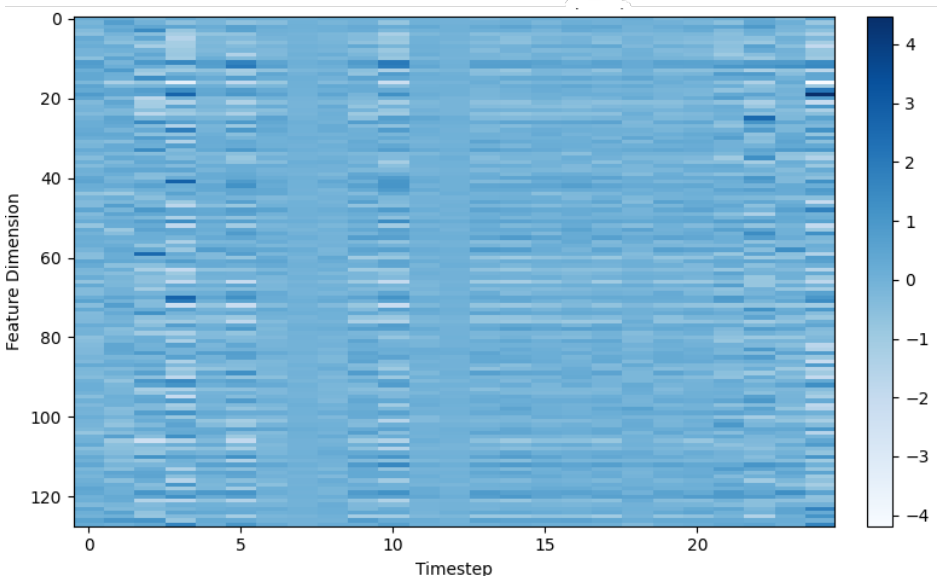


Figure 2: Raw activation values of final linear layer in first encoder block of a single test instance.

6.1 Training and Utilization of SAE

To train an SAE, we extract internal activations from our trained TST model, specifically from the final linear layer at the end of our first transformer encoder block. These activations represent the

internal computations of the TST model for an input timeseries. The SAE treats these activations as input data, and this is fundamentally different from training on raw time series; instead, the SAE learns a representation of the internal states of the transformer itself.

The SAE consists of an encoder network that maps high-dimensional activation vectors to a low-dimensional sparse code, and a decoder network that reconstructs the original activation from the sparse code. It is trained using mean squared reconstruction loss with an added sparsity regularization term:

$$\mathcal{L}_{SAE} = ||x - \hat{x}||_2^2 + \lambda \sum_j |z_j|$$

where x is the original activation, $\hat{x}$ is the reconstruction, z_j are components of the sparse code, and λ is the sparsity strength, and the L_1 regularization encourages most entries in z to be zero, pushing the network to represent inputs using only a few active latent features at a time. The introduced sparsity has the benefit of finding a low-dimensional basis over internal activations. Furthermore, it allows for the disentanglement of neurons as each sparse neuron tends to encode a distinct motif [CER+23], which in turn allows for easier interpretability.

Once trained, the SAE is applied to any test instance to extract these sparse activations. Thus, for a given SAE neuron and timestep, we find the top-k instances across the test set where its activation is the highest. We then inspect the corresponding raw time series inputs, check for their predicted and true class labels, and form potential hypothesis: "Neuron n consistently activates at timestep t when there is a peak for channel c"

To confirm a hypothesis, causal patching experiments could be entirely feasible. One could identify a high-activation SAE neuron, manually set that neuron to a high value in a target instance's sparse code z, decode the modified code into the transformer's original activation space, and patch this back into the model's MLP output. In the scope of this thesis, this was not done; however, such studies have been conducted on NLP tasks [CER+23].

7 Experiments & Results

This section walks through our experiments on a trained TST for one instance pair. Starting from model validation and instance selection, we gradually zoom in, from full layers to heads to individual timesteps, in order to identify where decisions emerge. For robustness, the entire analysis is conducted on a multitude of instance pairs across various datasets, and results can be found on GitHub [Kal25]. Additionally, an instance pair that boasts similar findings to the ones showcased below is included in Appendix A.

7.1 Baseline Performance and Instance Selection

Before applying any interpretability techniques, it is essential to confirm that our trained TST model achieves strong and consistent classification performance, allowing us to ensure that it has learned meaningful representations rather than guessing or memorizing noise, and that possible explanations derived from MI on this model reflect something insightful.

After training our TST model on the JapaneseVowels dataset as described in Section 4, the classifier achieves a test accuracy of 97.57%. The high accuracy shows that the model generalizes well to unseen examples despite the short time series and relatively small number of training samples per class. We visualize the predictions in a confusion matrix shown in Figure 3. Each row corresponds to the true class label, and each column to the predicted label. We see that only some classes have a few misclassification cases.

When it comes to interpretability, this is a highly desirable property; if the model were underperforming, it would lead to uncertainty whether causal patches or other analyses were reflecting the true signal or simply the model's inability to generalize and classify. In our case, the strong performance allows us to confidently say that the internal components are contributing to real and meaningful decisions. Moreover, the minor misclassification cases point us to edge cases, which we use for testing of intervention via activation patching.

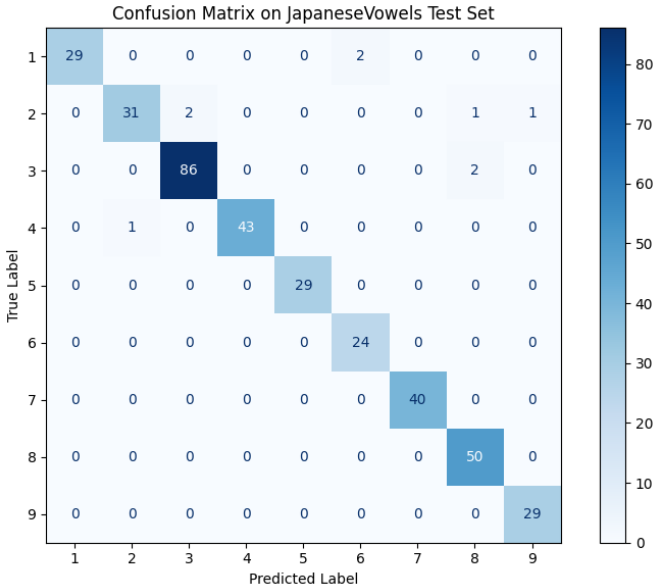


Figure 3: Confusion matrix of the trained TST on the JapaneseVowels test set.

Based on the above-mentioned and a deeper look at the few misclassified instances, for our activation patching experiments, we select the following instances depicted in Table 2. The pair is ideal for denoising-style activation patching as both belong to the same true class, and the internal activations are aligned in semantic space. The chosen instances' raw time series are visualized in Figure 1.

Instance ID	Type	True Class	Predicted Class	True Class Confidence
33	Clean	2	2	100.00%
44	Corrupt	2	3	10.72%

Table 2: Overview of selected instance pair for activation patching experiments

7.2 Layer-Level Causal Influence via Activation Patching

To begin our causal tracing of the model's internal decision-making, we start at the *coarsest* level of granularity; entire layers. Specifically, we patch attention heads in a given encoder layer of the corrupt instance with those from the clean instance. This allows us to see the overall influence of an entire layer's attention block on the model's output.

This forms the first step in a progressively more fine-grained investigation. We can identify if any layer is critical for the model's performance and decisions, or causes errors. The results are visualized in Figure 4. Each bar represents the change in the model's predicted probability for the true class after replacing all heads in one layer.

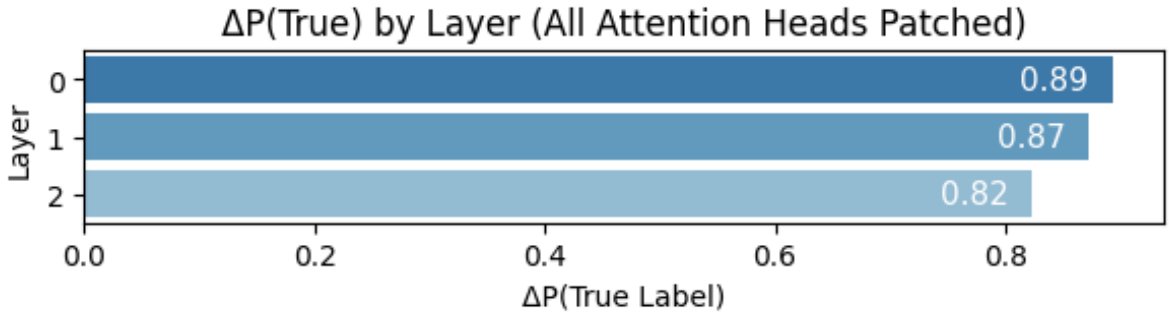


Figure 4: Layer-level activation patching results. Each bar represents the change in true class confidence (ΔP) when all attention heads in a given layer are patched from a clean instance.

Significant change in classification confidence (ΔP) suggests that the whole layers encodes important class-relevant computations. These high values across all three layers imply that the model's representation is distributed - each layer has meaningful contributions to the final decision. Nevertheless, a gradual decrease in ΔP is noticeable. This motivates us to perform subsequent experiments, in which we will narrow down to specific components within these layers that are most influential.

7.3 Head-Level Causal Influence via Activation Patching

We increase the granularity of our interventions by inspecting the influence of individual attention heads. Thus, instead of patching all heads in a layer simultaneously, we patch one head at a time.

The results of this sweep are shown in Figure 5. Each cell in the heatmap corresponds to a single attention head at a specific layer and represents the probability change for the true class after patching only that head.

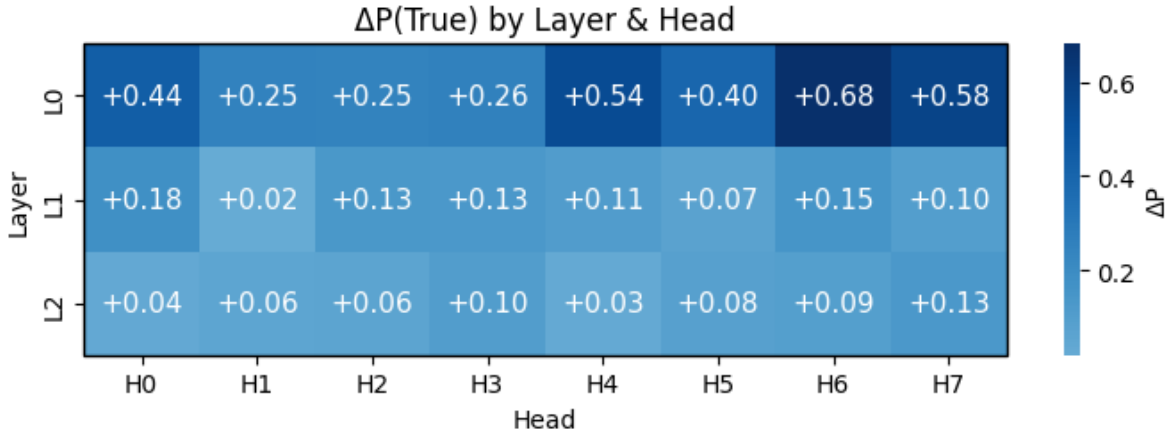


Figure 5: Heatmap showing head-level activation patching results. Each cell (Layer index, Head index) represents the probability change for the true class (ΔP) after patching a single attention head.

Several interesting points emerge. Namely, Layer 0 [2] now exhibits clear dominance in terms of head-level influence. Moreover, within Layer 0, patching heads 0, 4, 5, 6, and 7 individually is sufficient to restore correct classification of the corrupt instance (consider baseline confidence of 10.57%, thus in case of H6 patch, the model confidence becomes $10.57 + 68 = 78.57\%$).

On the other hand, layers 1 and 2 exhibit significantly lower influence. The influence across heads is also more evenly distributed, suggesting that they refine rather than drive the core decision-making of the model in this case. Lastly, although not clearly visible, layer 2 heads do show slightly less causal influence than layer 1, supporting the gradual decrease in layer influence showcased in Figure 4.

These findings demonstrate that causal influence is not uniform across all heads; rather, it is **concentrated** in a few key components. With this in mind, an even finer level of detail investigation is warranted.

7.4 Head × Position-Level Causal Inluence via Activation Patching

Now we can investigate the behavior of the attention heads by examining **timestep-specific** contributions. That is, we patch the output of a specific head at a position (timestep) and see how these most granular patches affect the model's predictions. This can answer a potential question: *which parts of the input does this head attend to causally?*

[2] Layer and head numbers are zero-indexed, so "Layer 0" refers to the first encoder layer.

Figure 6 showcases the effect for the most influential head determined in the previous step - Head 6 in Layer 0, visualizing the change in predicted probability for the true class after patching each position independently.

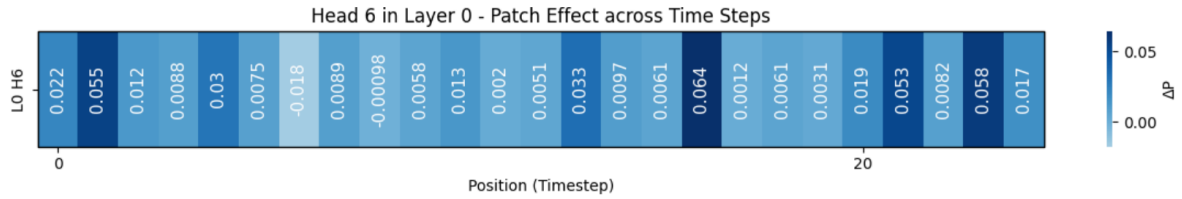


Figure 6: Heatmap showing position-level activation patching results on Head 6 in Layer 0. Each cell represents the probability change for the true class (ΔP) after patching that specific timestep in the chosen head.

From the plot, we notice that about half of the positions contribute very little to the classification, with ΔP values less than 0.01, and 5 more positions barely exceeding this threshold. Moreover, certain positions, namely 6 and 8, yield a negative ΔP value. Conversely, some specific positions show a more notable influence. Timesteps 1, 16, 21, and 23 all exceed 0.05 ΔP value. This result can potentially highlight not only which heads matter, but **when** they matter.

Here, we importantly note, the cumulative ΔP value of all position patches is

$$\sum_{t=0}^{T-1} \Delta P_t \approx 0.43,$$

whereas patching the full head gives a larger $\Delta P = 0.68$. This discrepancy raises the point that the influence of the head is **not** simply additive across timesteps. Rather, it reflects the presence of nonlinear interactions and contextual dependencies (the transformer architecture integrates information across positions via self-attention, residual connections, and layer normalization). As such, patching multiple positions together may have synergetic or (as later shown) interfering effects. This and other implications will be discussed in more detail in Section 8 and the following subsection.

7.5 Attention Saliency

To accompany the intervention technique *observations*, we can compare them with results from observational methods such as attention salience. It captures which input timesteps are attended to by a specific head, averaged across all queries in the sequence.

In Figure 7 we visualize the raw input timeseries for both the clean and corrupt instance, overlaid with saliency scores from Head 6 in Layer 0.

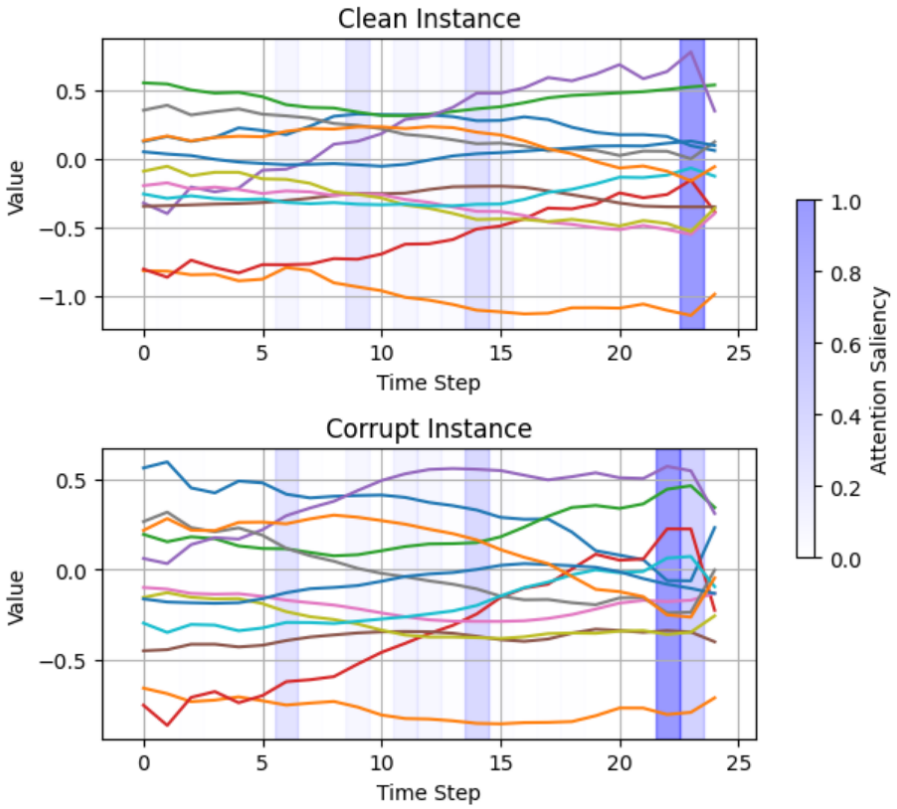


Figure 7: Attention saliency scores from Head 6 in Layer 0 overlaid on the raw input time series for the clean and corrupt instances

From the figure, we can draw some insights. For instance, we see that saliency is not distributed evenly across the sequence; rather, certain timesteps receive noticeably more attention. However, the notion that attention saliency would align with results observed in the patching experiments must be discarded. Attention saliency **does not** imply causality. This has been noted in prior work, particularly in the context of NLP, where attention distributions may act as routing mechanisms rather than attribution signals.

Nevertheless, attention saliency can be used as a useful diagnostic interpretability tool, helping identify candidate positions for more detailed causal intervention. In our case, the salient plots offer a plausible explanation for the head's behavior. The head appears to focus on a small number of timesteps in the same way that only a small number of timesteps have causal influence via patching. More discussion on this method will be done in Section 8.

7.6 Accumulating Top-k Critical Patches

To evaluate how influential components interact when combined, we conduct a controlled multi-patching experiment. Starting from the most influential (Layer, Head, Position) triplets, which were identified via individual ΔP scores, we sequentially apply patches from the clean instance into the corrupt instance.

For each $k \in \{1, \ldots, 10\}$, we apply the top-k critical patches and observe the resulting change in predicted probability for the true class. Table 3 summarizes the results.

Top-k Patches	ΔP(True)	P(True) **Final**
1	0.1270	0.2342
2	0.1716	0.2788
3	0.2058	0.3131
4	0.3055	0.4128
5	0.4324	0.5396
6	**0.3712**	**0.4784**
7	0.4502	0.5575
8	0.4835	0.5908
9	0.5772	0.6844
10	0.6375	0.7448

Table 3: Results of applying top-k critical patches. Shows cumulative ΔP and final predicted probability for the true class as more patches are applied.

The results show that the model's confidence increases steadily as more top-ranked patches are applied, but not monotonically. Notably, applying the 6th patch results in a **drop in confidence** compared to the 5-patch version, despite being ranked as a top contributor when evaluated in isolation.

This effect illustrates a key point already discussed previously: when multiple internal components are activated simultaneously, their influence may interfere, overlap, or even cancel out due to the model's nonlinear architecture. Components that are helpful alone may become redundant or conflicting when patched in tandem with others.

7.7 Building Causal Graphs from Patches

To better understand the structure of this influence, we construct a **causal graph** that represents the flow of information through the model. Each patch is treated as a directed edge from a source to a target node, with edge weights corresponding to observed changes in the true class confidence (ΔP). The graph is structured into three tiers. Namely, *input layer*, nodes representing individual timesteps from the input sequence (e.g., T 0, T 1, ..., T 24), followed by *transformer internals*, nodes representing attention heads at specific layers (e.g., L0H6 for Head 6 in Layer 0), and ending in *output Llayer*, nodes representing the model's class logits (e.g., C 2 for Class 2).

The previous experiment showed that patching the top-5 critical (Layer, Head, Position) combinations from a clean instance is sufficient to restore correct classification in the corrupt instance, achieving a final confidence $P(y_{\text{true}}) = 0.5396 > 0.5$. This strongly suggests that these components encode key information responsible for the correct decision. Figure 8 visualizes the resulting graph using the top-5 most influential patches. In this case, each edge from a time step node to a head node corresponds to a single patch, and the edge from a head node to the class node holds the same weight as the corresponding patch, simply signifying the influence towards prediciting the true class.

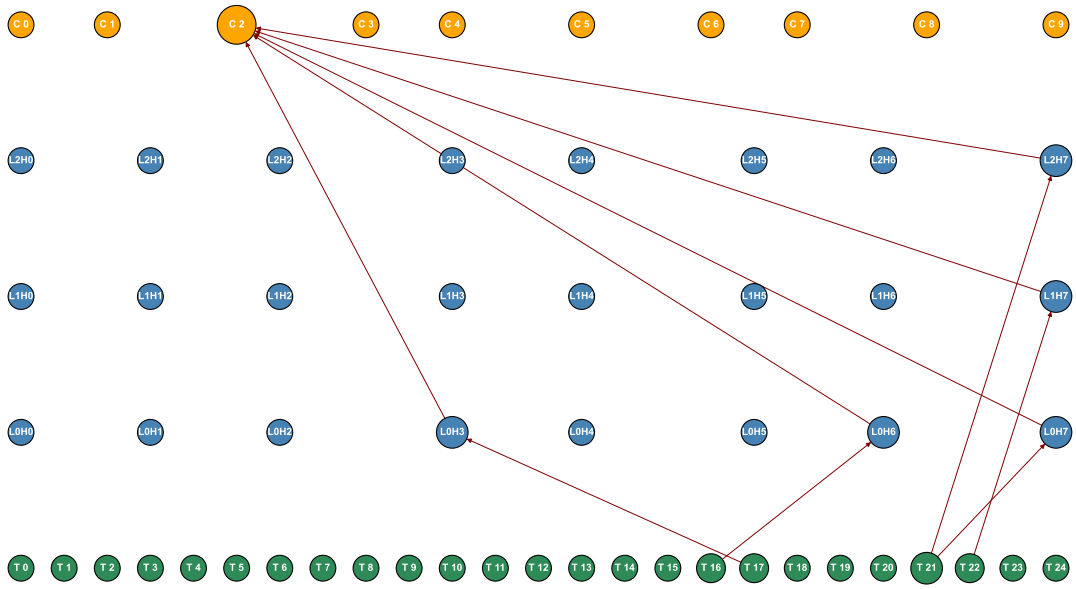


Figure 8: Causal graph showing top-5 most influential patches connecting input timesteps, internal attention heads, and the final class output.

The five patches involve five different attention heads and four unique input time steps, with Time 21 appearing twice. Most of the heads (three out of five) are located in Layer 0, while the remaining two come from Layers 1 and 2, respectively. Table 4 lists the patches in descending order of ΔP. Interestingly, the top three patches all target Head 7 in Layers 0, 1, and 2, suggesting that the final head in each layer may play a particularly significant role in encoding the relevant decision information for this instance.

Rank	**Time Step Node**	**Head Node**	ΔP
1	T 21	L0H7	0.127
2	T 21	L1H7	0.087
3	T 22	L2H7	0.081
4	T 16	L0H6	0.064
5	T 17	L0H3	0.062

Table 4: Top-5 most influential patches ranked by ΔP

While the top-5 graph highlights the most influential individual patches, it only captures a small subset of the model's internal structure. To get a broader view of the information flow, we apply a threshold-based filtering approach. Instead of selecting a fixed number of top-ranked patches, we include all patches whose causal effect exceeds a set ΔP threshold. Specifically, we include Head $\rightarrow$ Class edges with $\Delta P \geq 0.1$ and Timestep $\rightarrow$ Head edges with $\Delta P \geq 0.01$, but only if the corresponding head was influential enough to reach the class node.

Figure 9 shows the resulting graph. Unlike the sparse and targeted view in Figure 8, this version is denser and reveals additional components that contribute moderately but were excluded

from the top-5. The structure now includes heads that aggregate signals from multiple time steps and shows more branching from the input layer. While less interpretable at a glance, this view gives a more complete picture of how influence is distributed across the model.

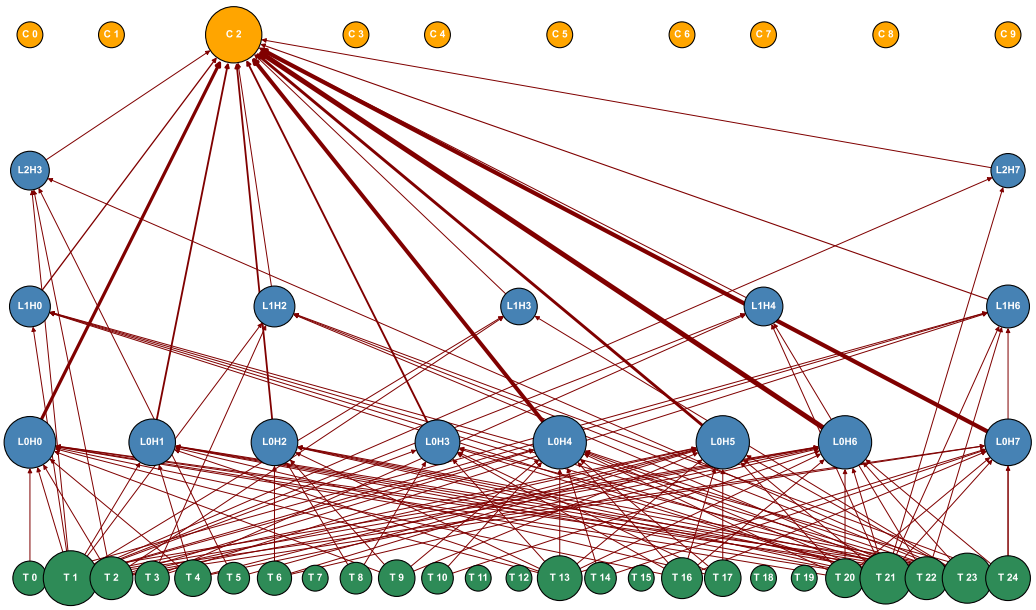


Figure 9: Causal graph showing patches that exceed the ΔP threshold. Head $\rightarrow$ Class threshold is 0.1, edges Timestep $\rightarrow$ Head threshold is 0.01, and only show edges to Head nodes that met the Head $\rightarrow$ Class threshold.

To simplify this view, we compute the node degrees for each component in the graph. Specifically, we count how many outgoing edges each timestep node has (i.e., how many heads it connects to), and how many incoming edges each head node receives from timesteps (i.e., how many inputs it draws from). We exclude Head $\rightarrow$ Head edges from this analysis to focus purely on the flow from input to internal structure.

Figure 10 shows the result. On the input side, we note that among the most influential time steps, four of them, Time 21, 22, 23, and 24, are located at the tail end of the sequence. Interestingly, Time 1 also ranks highly, whereas Time 0, the first timestep, does not. This suggests that early sequence information is not uniformly important, and the model may rely more heavily on later input for this particular prediction. On the attention head side, we observe that out of the top 10 highest degree heads, 8 belong to Layer 0. This is consistent with our earlier results showing that patching Layer 0 heads had the highest impact on the model's predictions (see Figure 4), at least for our selected instances.

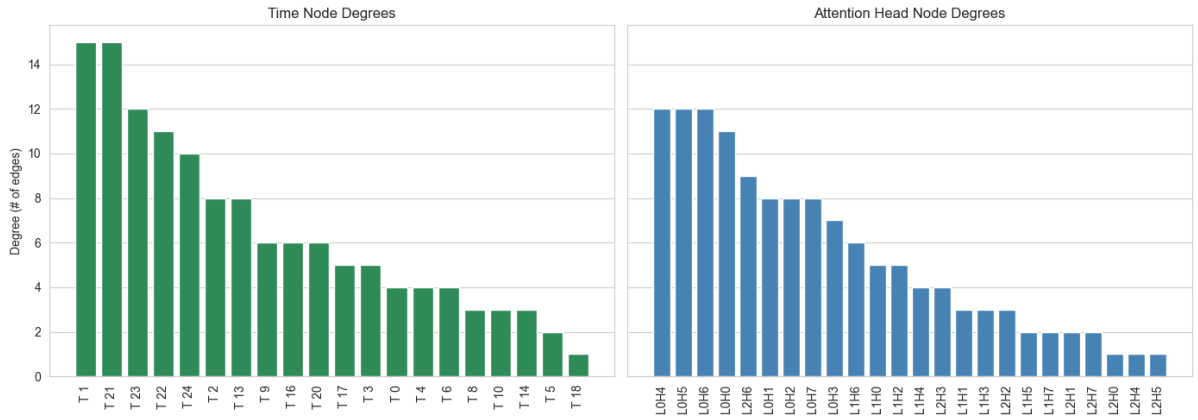


Figure 10: Node degree visualization from the causal graph in Figure 9. [Left] Degree of Timestep nodes *(excluding Head → Class edges)*. [Right] Degree of Head nodes.

7.8 Provisional Results from Sparse Auto Encoders

To explore whether sparse autoencoders can yield semantically meaningful and selective features, we trained an SAE on the internal activations of the TST model, specifically from the output of the MLP block (`linear2`) at a first encoder layer. After training, the encoder provides sparse representations of shape (N, T, H), where H is the number of SAE neurons, T is the number of timesteps, and N is the number of instances. These sparse codes reveal when and where each neuron activates, allowing us to identify patterns and potential class-selective features.

Figure 11 shows an illustrative example of this analysis. The neuron of interest here is SAE neuron 15, which shows a high activation at timestep 22 in multiple test instances. All top-activating instances correspond to Class 8, suggesting that this neuron might encode a feature associated with that class.

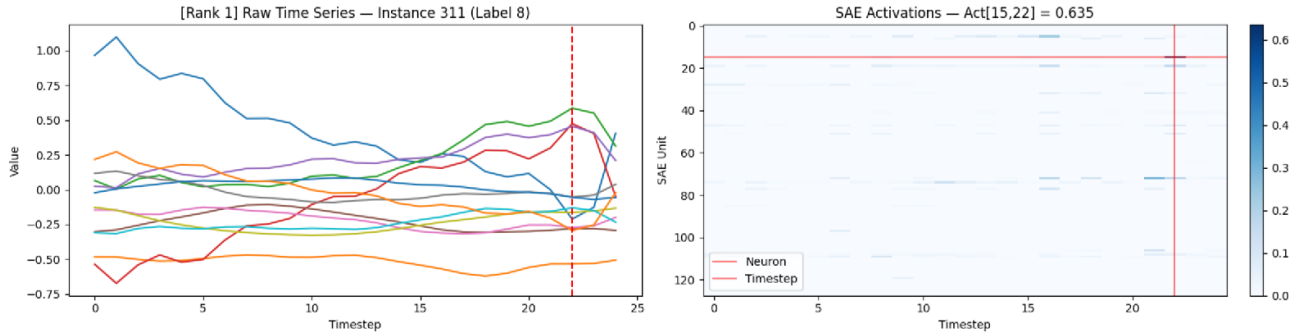
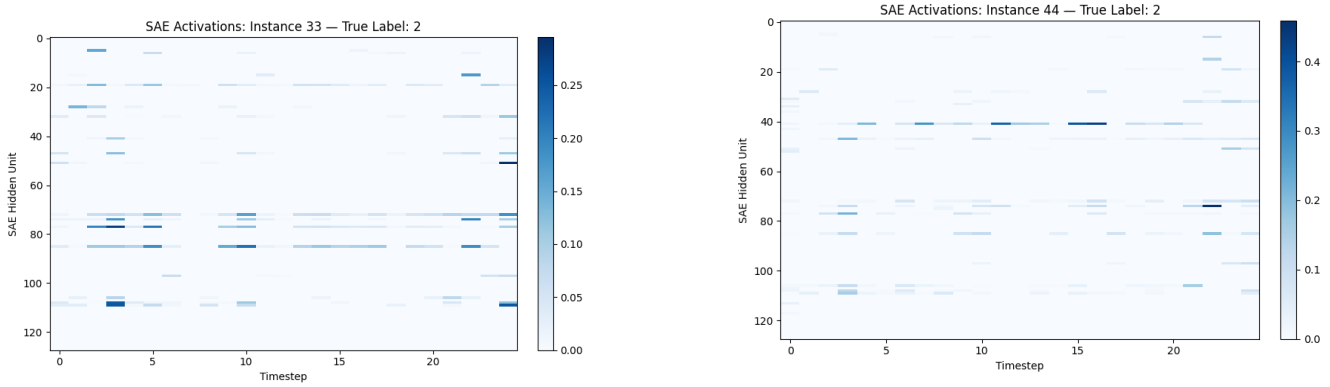


Figure 11: [Left] Raw time series input for the top-activating instance of SAE neuron 15 at timestep 22. [Right] Corresponding SAE activations, highlighting Neuron 15 at timestep 22.

A consistent pattern is visible in the raw input: each high-activation instance exhibits a peak in Channels 2 (green), 5 (red), and 9 (purple) and a dip in Channel 7 (blue) around Timestep 22. This repeated signal structure could suggest that SAE neuron 15 has learned to represent this particular pattern, which might be a class-selective temporal motif. Examples to substantiate this

hypothesis are provided in the Appendix B, Figure 22, where we visualize the top-k activations for this particular neuron.

Beyond analyzing isolated neuron-timestep activations, we also investigate how the sparse activations look in our chosen clean and corrupt instances. The aim is to discover whether sparse neurons exhibit class-selective activation patterns, which may point to their potential as meaningful, disentangled features. Figure 12 visualizes the sparse codes for the two instances.



(a) Instance 33

(b) Instance 44

Figure 12: Sparse autoencoder activation heatmaps for two test instances. [Left] A clean instance correctly classified as Class 2. [Right] A corrupt instance misclassified as Class 3. Each heatmap shows SAE neuron activations over timesteps.

In Figure 12b, we observe that neuron 41 shows strong activation across several timesteps. We find that neuron 41 is frequently active in Class 3 instances. On the other hand, the sparse codes seen in Figure 12a show stronger activations in neurons 85 and 78, with additional scattered activity across other units. This activation pattern is more consistent with what we observe across the broader set of correctly classified Class 2 instances. In Appendix B Figure 23, we provide further visualizations showing the activation of these neurons across other Class 2 and Class 3 test samples. Furthermore, in Appendix B, other notable SAE neurons are listed.

These observations support the idea that SAE neurons are capturing class-discriminative features, albeit in a sparse and distributed manner. Misclassifications may occur in part when a corrupt instance inadvertently activates features more representative of another class, which could be further explored through targeted patching. It is important to emphasize that SAE neurons do not correspond one-to-one with raw TST neurons. That is, SAE neuron 15 does not represent TST's neuron 15 in `linear2`. Instead, each SAE neuron encodes a distributed feature, a specific direction in the transformer's latent space learned during training. Therefore, the procedure for patching would be the following. First, encode activations from the TST into sparse code z using the SAE encoder. Then manually boost the activation of the selected sparse neuron, e.g., $z_i = 5$. Followed by decoding this modified z back to $\hat{x}_{\text{patched}}$ using the SAE decoder. Replace the original TST activation with the $\hat{x}_{\text{patched}}$ at the chosen layer and timestep. This can then determine the causal influence of a single sparse feature on the model's decision.

8 Discussion

Interpretability is highlighted as a cornerstone of AI safety [BG24] in numerous high-stakes domains. As AI models grow ever more complex and are being deployed on a scale never seen before, the understanding of their internal decision-making processes is critical. The extension of MI to time series classification is not merely beneficial, but rather **essential**.

A time series transformer model's predictions can have substantial practical and ethical implications. In most critical sectors, namely healthcare and finance, the understanding of the causal influence of internal components can aid in trust-building, risk mitigation, and ethical alignment.

The experiments have provided compelling evidence that mechanistic interpretability methods, particularly activation patching, can be successfully adapted from NLP to time series transformer models. There is potential for deeper interpretability in time series modeling, especially in determining the causal influence of internal transformer components. Our findings align with similar discoveries in text [MBAB23, WVC+22] and vision [Bah25] domains, highlighting the importance of selective component analysis within transformer architectures. This alignment not only shows the effectiveness of the chosen methods across domains but also emphasizes the importance of a systematic approach to interpretability.

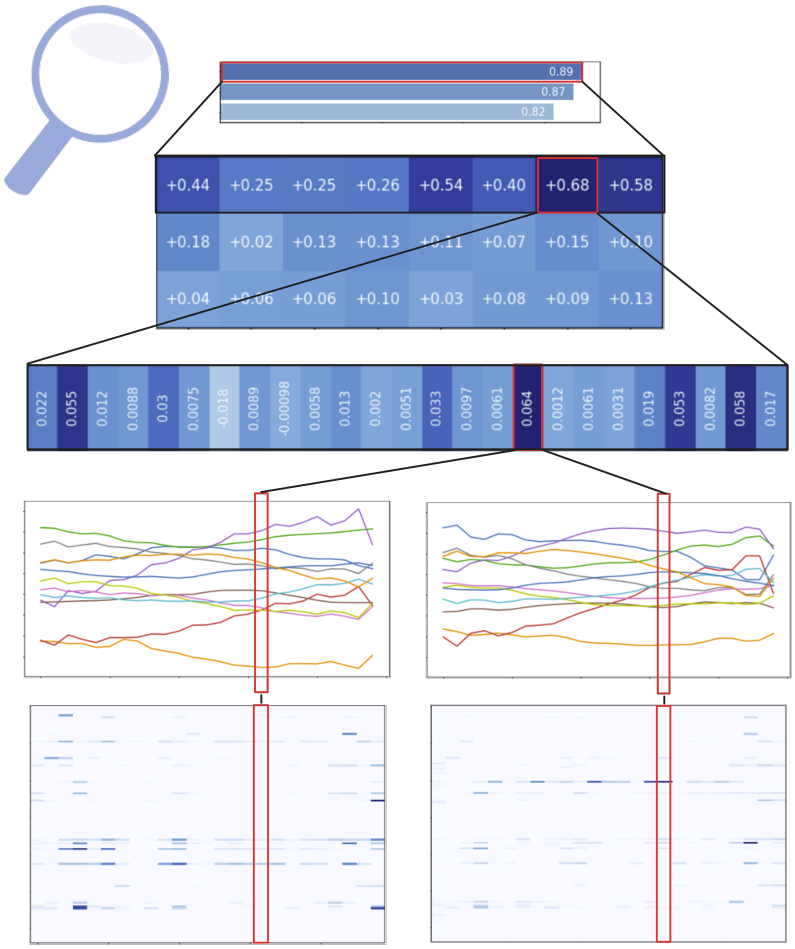


Figure 13: Illustration of multilevel interpretability in a Time Series Transformer

8.1 Comparison to Traditional Post-Hoc Methods

Traditional explainers such as SHAP and LIME operate only at the input–output level. They tell us which timesteps or channels matter, but not which internal components create that importance. MI tools extend this capability by testing causal necessity and sufficiency inside the network. An overview of the qualitative comparison between traditional post-hoc and MI methods is presented in Table 5.

Aspect	**SHAP / LIME**	**Mechanistic Interpretability**
Scope	Model-agnostic	Model-specific
Internals Needed	None	Activations, hooks
Target	Input features	Internal components
Mechanism	Correlational (via perturbation)	Causal (via intervention)
Causal Validity	Approximate	Directly tested
Robustness	Partial	Sensitive to entanglement
Computational Cost	Low	High
Scalability	Linear	Combinatorial
Output	Saliency maps, scores	Patch effects, causal graphs
Use Case	Sanity checks, explainability	Root cause, model editing
Tooling	Mature libraries	Early-stage, custom scripts
Accessibility	Easy to interpret	Technical, harder to communicate
Stability	Stable	Sensitive to design

Table 5: Compact comparison of post-hoc attribution vs. mechanistic interpretability in TSTs

One of the key distinctions is that SHAP works on any model, regardless of architecture or internals, it requires only input–output access. In contrast, MI requires access to internal activations, making it limited to models where hooks or modifications are possible. Furthermore, SHAP produces easily visualizable saliency maps, whereas MI yields causal graphs or patch effects that are harder to interpret and require custom scripts. Moreover, while post-hoc methods scale well and offer stable outputs, MI methods are more sensitive to entangled representations and patch configuration. Nevertheless, MI remains uniquely suited for uncovering root causes and enabling model edits - capabilities that justify its added complexity.

In practice, a hybrid workflow could be useful: first use SHAP to shortlist influential timesteps, then apply targeted patching to test which components respond causally. This is particularly valuable for long real-world time series, where exhaustive patching becomes a monumental task. On short sequences like JapaneseVowels, we can patch all positions, but SHAP can be used as a filtering tool, which could prove essential for scaling MI to large-scale settings.

8.2 Limitations of Our Work

Despite the insights highlighted above, current MI methods require extensive manual effort, from careful selection of instances, determination of granularity, and choosing of the most suitable approach, to actual *interpretation* of the results and construction of causal graphs, pseudocode, or any other human-interpretable conveyance method. All of these processes together demand expertise and meticulous experimentation. Consequently, one may face substantial bottlenecks.

In this thesis, we have developed a foundational pipeline; however, it currently addresses only **one** component - attention heads. Other significant components, such as MLP layers, residual streams, positional embeddings, or individual neuron activations, were not integrated into our experiments. Nevertheless, each of these components could potentially hold meaningful insights into model decisions. Therefore, expanding and refining the pipeline to be comprehensive is necessary.

Following this, the interpretation of the results in terms of determining learned features in time series contexts raises a unique challenge. Unlike NLP, where features correlate to semantic concepts, syntax, or specific words, time series features lack straightforward semantic definitions. Some meaningful interpretations could involve recognizing temporal patterns, such as peaks, valleys, periodic behaviors, transitions, domain-specific events, or anomalies. Several questions arise when interpreting results achieved via patching. Do the potential temporal concepts uncovered correspond to the specific instance, or the dataset, or a broader domain, or to general time series features? This question connects to the notion of **universality** [CLJ23], which suggests that neural networks trained on similar tasks provide the emergence of common representations [SMW+24] [KNLH19]. That is, rather than each model learning arbitrary features, there is an underlying universality to the circuits that emerge in a network, which is shaped by the task and certain biases. Another interpretation can be that, given a task, dataset, and model architecture, there may simply exist an optimal way to solve the problem, and different models converge towards it [BG24]. Whether temporal patterns discovered through MI in time series reflect generalizable temporal features or remain context-specific is an open question.

Additionally, interpreting results from high-dimensional time series data can be a daunting task. With dimensions often reaching hundreds, making sense of the "mess" raises some practical questions. How to systematically determine which dimension is playing the key role? Can dimensionality reduction techniques simplify interpretations without losing critical information?

Lastly, as our multi-patching experiments revealed, the interactions within a transformer architecture are complex. Critical insights are not revealed straightforwardly; instead, interactions between model components can yield unpredictable outcomes. Specifically, interactions may sometimes enhance each other's effects, resulting in a stronger combined influence than individual components. Conversely, they might diminish each other's contributions, thereby negating the individual benefits previously observed when analyzed separately. There are also instances where combinations of patches exhibit minimal or no effect.

One possible factor contributing to the non-additivity of $\sum_{t=0}^{T} \Delta P_t$ is the distribution of attention itself. Attention saliency, as visualized in Figure 7, indicates that a few specific timesteps receive a disproportionately large amount of attention from Head 6 in Layer 0. These positions are therefore referenced by many query positions within the sequence, meaning that a change at one position can affect multiple outputs. However, when positions are patched individually, these cross-position effects are not fully captured due to the sequential and localized nature of the intervention. In contrast, patching the entire head simultaneously preserves the structure of the attention matrix and allows influential positions to act in concert.

This complexity is primarily driven by the inherent characteristics of transformer architectures. The simultaneous functioning of multi-head attention, residual connections, and layer normalizations introduces numerous interdependencies within the model. These architectural components facilitate complex internal computations that allow transformers to effectively model intricate temporal and contextual dependencies. However, this necessitates developing more advanced MI frameworks capable of explicitly modeling such complexities.

8.3 Future Directions

Regarding future improvements, broadening the scope of the patching experiments is the main one. The patching experiments highlighted and analyzed in this thesis use a single clean-corrupt pair. A more comprehensive analysis would involve systematically repeating these experiments across many instance pairs and class combinations. This could help establish class-specific or instance-agnostic causal components and determine if certain attention heads or neurons are generally responsible for certain behaviors across the dataset.

Furthermore, the use of the JapaneseVowels dataset allowed rapid iteration and controlled analysis. Future research should apply these MI techniques to other datasets, including higher-dimensional, irregular, or real-world time series. Testing on different transformer variants would also provide insight into whether the discovered causal patterns generalize across architectures.

Additionally, the building of causal graphs could be improved by more sophisticated search strategies like evolutionary algorithms, which could be used to identify minimal, non-redundant sets of patches that maximize impact. This aligns with emerging interest in combining MI with optimization techniques to uncover minimal causal circuits. Mayhaps, in the search for better causal graphs, future studies may also consider alternate or complementary metrics, as ΔP served as a simple causal metric. Then, as causal graphs grow in size and complexity, interpreting and communicating them to non-experts becomes harder. Future work may explore how to summarize, cluster, or narrate causal pathways using explainable AI principles.

Lastly, given that patching can correct model behavior, future research might explore using MI techniques as a post-hoc error correction mechanism. By identifying causal components of incorrect predictions, one could devise automated procedures to patch activations during inference, potentially increasing robustness without retraining.

9 Conclusion

This thesis set out to answer the central question: **Can mechanistic interpretability techniques developed for NLP transformer models be meaningfully adapted to the domain of time series classification ?** Through a careful transfer and experimental evaluation of the key methods on a trained Time Series Transformer model for the JapaneseVowels dataset, we find strong evidence that the answer is: yes, they can.

By adapting the notion of clean and corrupt instance pairs and performing structured denoising-style activation patching, we demonstrated that individual attention heads and even timestep-specific components within those heads can exert a measurable and often decisive causal influence on the model's predictions.

Causal graphs constructed from these patching experiments showed interpretable pathways from input timesteps through specific heads to class outputs, mirroring similar circuit-tracing approaches in NLP. Notably, we also observed nonlinear interactions and diminishing returns when combining top-k patches, highlighting the complexity of distributed computation in TSTs.

Alongside patching, Sparse Autoencoders and attention saliency techniques provided complementary visualizations, helping highlight which timesteps may be influential and thereby guiding hypotheses. Moreover, Sparse autoencoders, though explored more briefly, showed promise in extracting disentangled latent features that correlate with class-discriminative patterns in the data.

While our results validate the technical viability of adapting MI tools to time series transformers, we do not claim to have fully reverse-engineered our TST model. Instead, this thesis establishes a proof-of-concept that opens the door to future work in richer interpretability pipelines for time series models. This includes extending patching to residual streams and MLPs, exploring instance-agnostic circuits, and deploying evolutionary search or optimization techniques to refine causal graphs.

In sum, the methods developed for probing the inner workings of LLMs can illuminate the hidden logic of TSTs as well. With continued refinement, these tools hold the potential to bring not only interpretability but also accountability and robustness to deep time series models in critical domains.

References

[APSB15] Sahar Akram, Alessandro Presacco, Shihab Shamma, and Behtash Babadi. Robust decoding of selective auditory attention from meg in a competing-speaker environment via state-space modeling. *NeuroImage*, 124, October 2015.

[Bah25] Nooshin Bahador. Mechanistic interpretability of fine-tuned vision transformers on distorted images: Decoding attention head behavior for transparent and trustworthy ai, 2025.

[BCE+23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[BDL+18] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018, 2018.

[BG24] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024.

[CER+23] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[CES+24] Stephen Casper, Carson Ezell, Charlotte Siegmann, Noam Kolt, Taylor Lynn Curtis, Benjamin Bucknall, Andreas Haupt, Kevin Wei, Jérémy Scheurer, Marius Hobbhahn, Lee Sharkey, Satyapriya Krishna, Marvin Von Hagen, Silas Alberti, Alan Chan, Qinyi Sun, Michael Gerovitch, David Bau, Max Tegmark, David Krueger, and Dylan Hadfield-Menell. Black-box access is insufficient for rigorous ai audits. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 2254–2272. ACM, June 2024.

[CKLM19] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019.

[CLJ23] Lawrence Chan, Leon Lang, and Erik Jenner. Natural abstractions: Key claims, theorems, and critiques. AI Alignment Forum post, March 2023.

[DPW20] Angus Dempster, François Petitjean, and Geoffrey I. Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, July 2020.

[ENO+21] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework

for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[FSBCj24] Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. A primer on the inner workings of transformer-based language models, 2024.

[Gro19] Joachim Gross. Magnetoencephalography in cognitive neuroscience: A primer. *Neuron*, 104(2):189–204, 2019.

[GYS24] Xinyue Gu, Linxiao Yang, and Liang Sun. Explainable artificial intelligence for time series: A systematic survey, 06 2024.

[HGTH23] Keigo Hikita, Jose Gomez-Tames, and Akimasa Hirata. Mapping brain motor functions using transcranial magnetic stimulation with a volume conductor model and electrophysiological experiments. *Brain Sciences*, 13(1), 2023.

[HN24] Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024.

[IVS+24] Md Khairul Islam, Tyler Valentine, Timothy Joowon Sue, Ayush Karmacharya, Luke Neil Benham, Zhengguang Wang, Kingsley Kim, and Judy Fox. Interpreting time series transformer models and sensitivity analysis of population age groups to covid-19 infections, 2024.

[JOC+23] Vytene Janiukstyte, Thomas Owen, Umair Chaudhary, Beate Diehl, Louis Lemieux, John Duncan, Jane de Tisi, Yujiang Wang, and Peter Taylor. Normative brain mapping using scalp eeg and potential clinical application. April 2023.

[JQC+25] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Lukas Vierling, Donghai Hong, Jiayi Zhou, Zhaowei Zhang, Fanzhi Zeng, Juntao Dai, Xuehai Pan, Kwan Yee Ng, Aidan O'Gara, Hua Xu, Brian Tse, Jie Fu, Stephen McAleer, Yaodong Yang, Yizhou Wang, Song-Chun Zhu, Yike Guo, and Wen Gao. Ai alignment: A comprehensive survey, 2025.

[JSH+25] Sonia Joseph, Praneet Suresh, Lorenz Hufe, Edward Stevinson, Robert Graham, Yash Vadi, Danilo Bzdok, Sebastian Lapuschkin, Lee Sharkey, and Blake Aaron Richards. Prisma: An open source toolkit for mechanistic interpretability in vision and video, 2025.

[Kal25] Matiss Kalnare. Tst-mechanistic-interpretability: Causal analysis and attribution for time series transformers. https://github.com/mathiisk/TST-Mechanistic-Interpretability, 2025. GitHub repository.

[KNLH19] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019.

[KTS99] Masanori Kudo, Jun Toyama, and Masashi Shimbo. Japanese Vowels [dataset]. UCI Machine Learning Repository, 1999.

[LGA+25] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

[LL17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[LP10] Richard B. Lipton and Starr H. Pearlman. Transcranial magnetic simulation in the treatment of migraine. *Neurotherapeutics*, 7(2):204–212, 2010.

[LSK+19] Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: Canonical time-series characteristics, 2019.

[MBAB23] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[MWG+24] Faisal Mushtaq, Dominik Welke, Anne Gallagher, Yuri Pavlov, Layla Kouara, Jorge Bosch, Jasper van den Bosch, Mahnaz Arvaneh, Amy Bland, Maximilien Chaumon, Cornelius Borck, Xun He, Steven Luck, Maro Machizawa, Cyril Pernet, Aina Puce, Sidney Segalowitz, Christine Rogers, Muhammad Awais, and V S-So. One hundred years of eeg for brain and behaviour research. *Nature Human Behaviour*, 8, August 2024.

[RHCHM23] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks, 2023.

[RPF+21] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable artificial intelligence (xai) on timeseries data: A survey, 2021.

[RRAG11] José Romero, Deisy Ramirez, Linda Aglio, and Laverne Gugino. Brain mapping using transcranial magnetic stimulation. *Neurosurgery clinics of North America*, 22:141–52, vii, April 2011.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[SMW+24] Ilia Sucholutsky, Lukas Muttenthaler, Adrian Weller, Andi Peng, Andreea Bobu, Been Kim, Bradley C. Love, Christopher J. Cueva, Erin Grant, Iris Groen, Jascha Achterberg, Joshua B. Tenenbaum, Katherine M. Collins, Katherine L. Hermann, Kerem Oktar, Klaus Greff, Martin N. Hebart, Nathan Cloos, Nikolaus Kriegeskorte, Nori Jacoby, Qiuyi Zhang, Raja Marjieh, Robert Geirhos, Sherol Chen, Simon Kornblith, Sunayana Rane, Talia Konkle, Thomas P. O'Connell, Thomas Unterthiner, Andrew K.

Lampinen, Klaus-Robert Müller, Mariya Toneva, and Thomas L. Griffiths. Getting aligned on representational alignment, 2024.

[SS19] Sofia Serrano and Noah A. Smith. Is attention interpretable?, 2019.

[TCM+24] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

[TPT+25] Nicholas T. Trapp, Anthony Purgianto, Joseph J. Taylor, Manpreet K. Singh, Lindsay M. Oberman, Brian J. Mickey, Nagy A. Youssef, Daniela Solzbacher, Benjamin Zebley, Laura Y. Cabrera, Susan Conroy, Mario Cristancho, Jackson R. Richards, Michael J. Flood, Tracy Barbour, Daniel M. Blumberger, Stephan F. Taylor, David Feifel, Irving M. Reti, Shawn M. McClintock, Sarah H. Lisanby, and Mustafa M. Husain. Consensus review and considerations on tms to treat depression: A comprehensive update endorsed by the national network of depression centers, the clinical tms society, and the international federation of clinical neurophysiology. *Clinical Neurophysiology*, 170:206–233, 2025.

[WVC+22] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

[WZZ+23] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2023.

[ZJP+21] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2114–2124, New York, NY, USA, 2021. Association for Computing Machinery.

A Additional Instance Pair Analysis

Instance ID	Type	True Class	Predicted Class	True Class Confidence
58	Clean	2	2	100.00%
31	Corrupt	2	9	11.36%

Table 6: Overview of selected instance pair for activation patching experiments

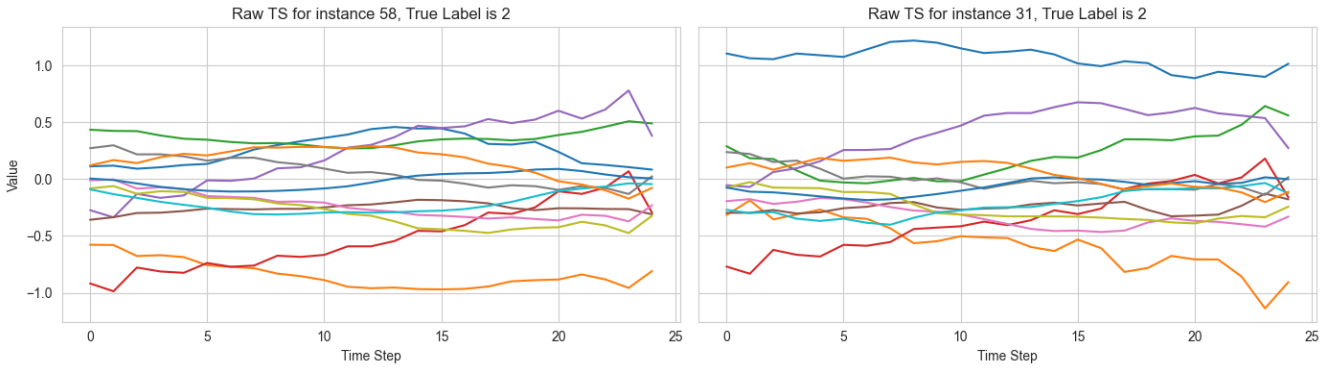


Figure 14: Raw time series for the two chosen instances. [Left] Clean Instance. [Right] Corrupt Instance

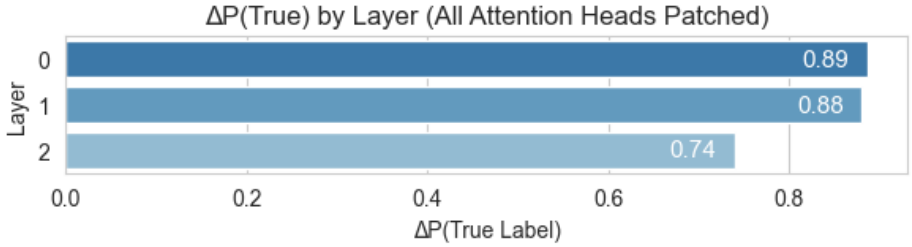


Figure 15: Layer-level activation patching results. Each bar represents the change in true class confidence (ΔP) when all attention heads in a given layer are patched from a clean instance.

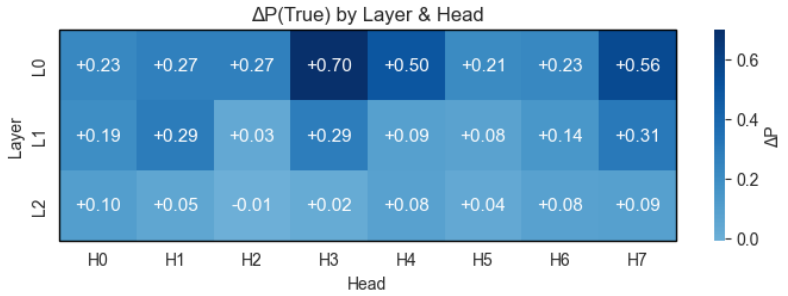


Figure 16: Heatmap showing head-level activation patching results.

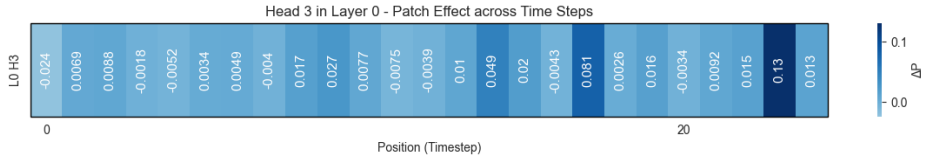


Figure 17: Heatmap showing position-level activation patching results on Head 3 in Layer 0.

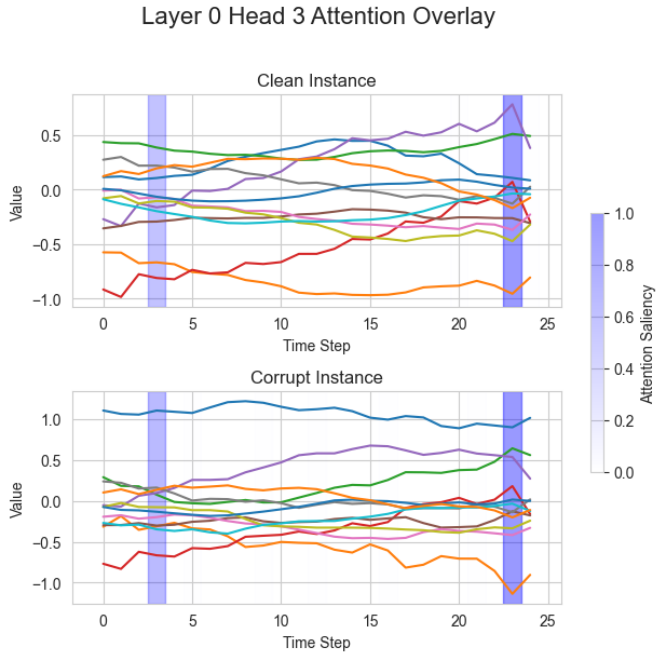


Figure 18: Attention saliency scores from Head 3 in Layer 0 overlaid on the raw input time series for the clean and corrupt instances.

Top-k Patches	ΔP(True)	P(True) **Final**
1	0.2248	0.3384
2	0.4557	0.5693
3	0.6435	0.7571
4	0.7108	0.8244
5	0.7217	0.8353
6	0.7860	0.8996
7	0.8209	0.9345
8	0.8445	0.9581
9	0.8513	0.9649
10	0.0518	0.9654

Table 7: Results of applying top-k critical patches. Shows cumulative ΔP and final predicted probability for the true class as more patches are applied.

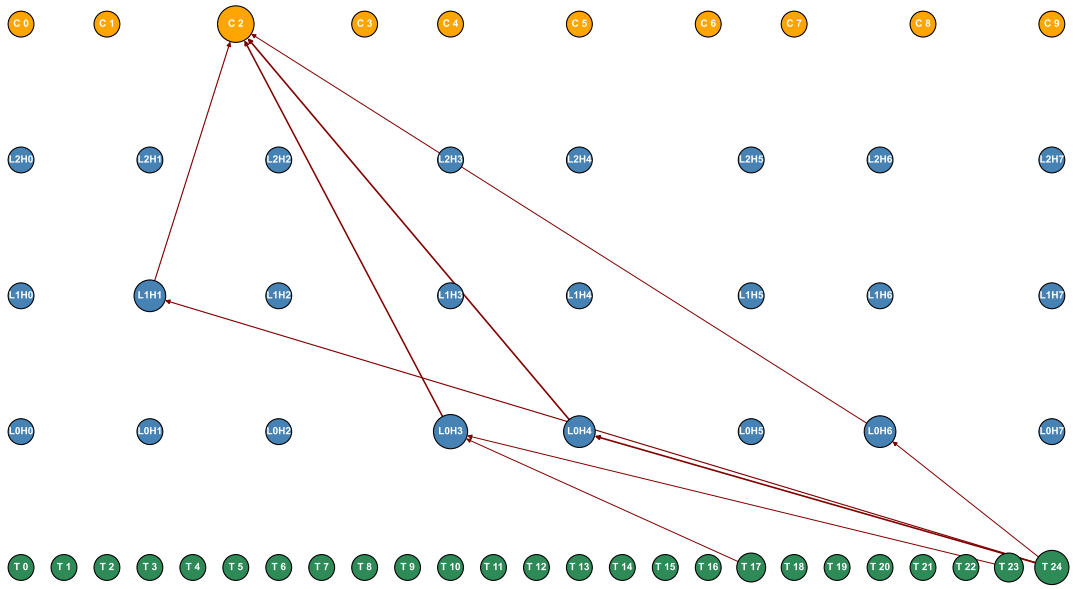


Figure 19: Causal graph showing top-5 most influential patches connecting input timesteps, internal attention heads, and the final class output.

Rank	**Time Step Node**	**Head Node**	ΔP
1	T 24	L0H4	0.225
2	T 24	L1H1	0.155
3	T 23	L0H3	0.130
4	T 17	L0H3	0.081
5	T 24	L0H6	0.077

Table 8: Top-5 most influential patches ranked by ΔP

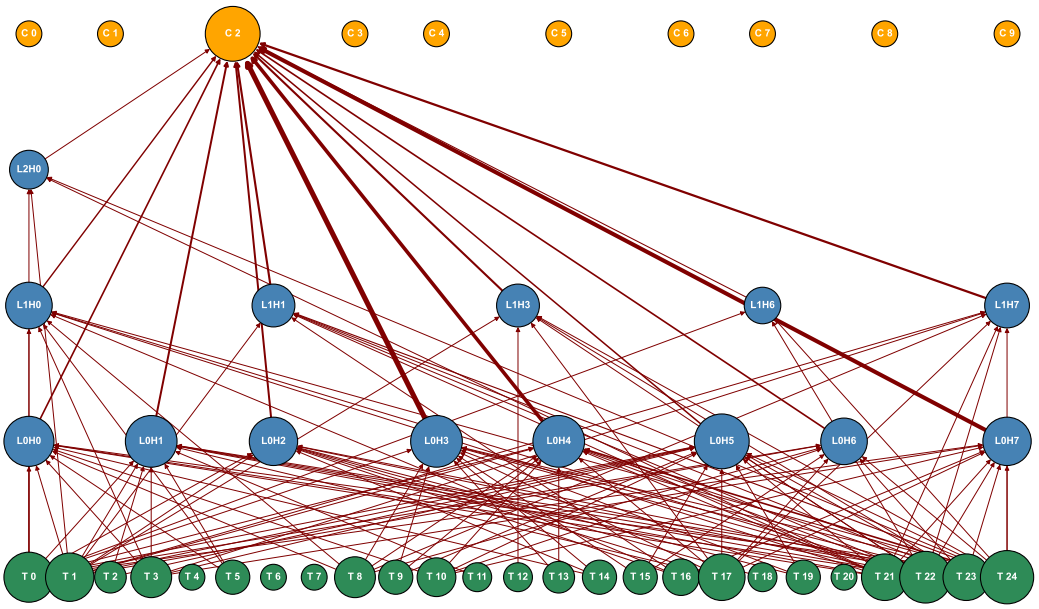


Figure 20: Causal graph showing patches that exceed the ΔP threshold. Head $\rightarrow$ Class threshold is 0.1, edges Timestep $\rightarrow$ Head threshold is 0.01, and only show edges to Head nodes that met the Head $\rightarrow$ Class threshold.

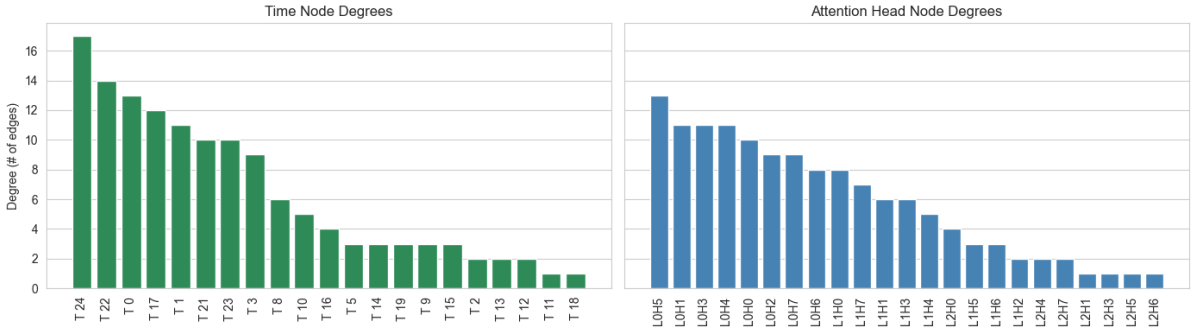


Figure 21: Node degree visualization from the causal graph in Figure 9. [Left] Degree of Timestep nodes *(excluding Head $\rightarrow$ Class edges)*. [Right] Degree of Head nodes.

B Additional SAE Observations

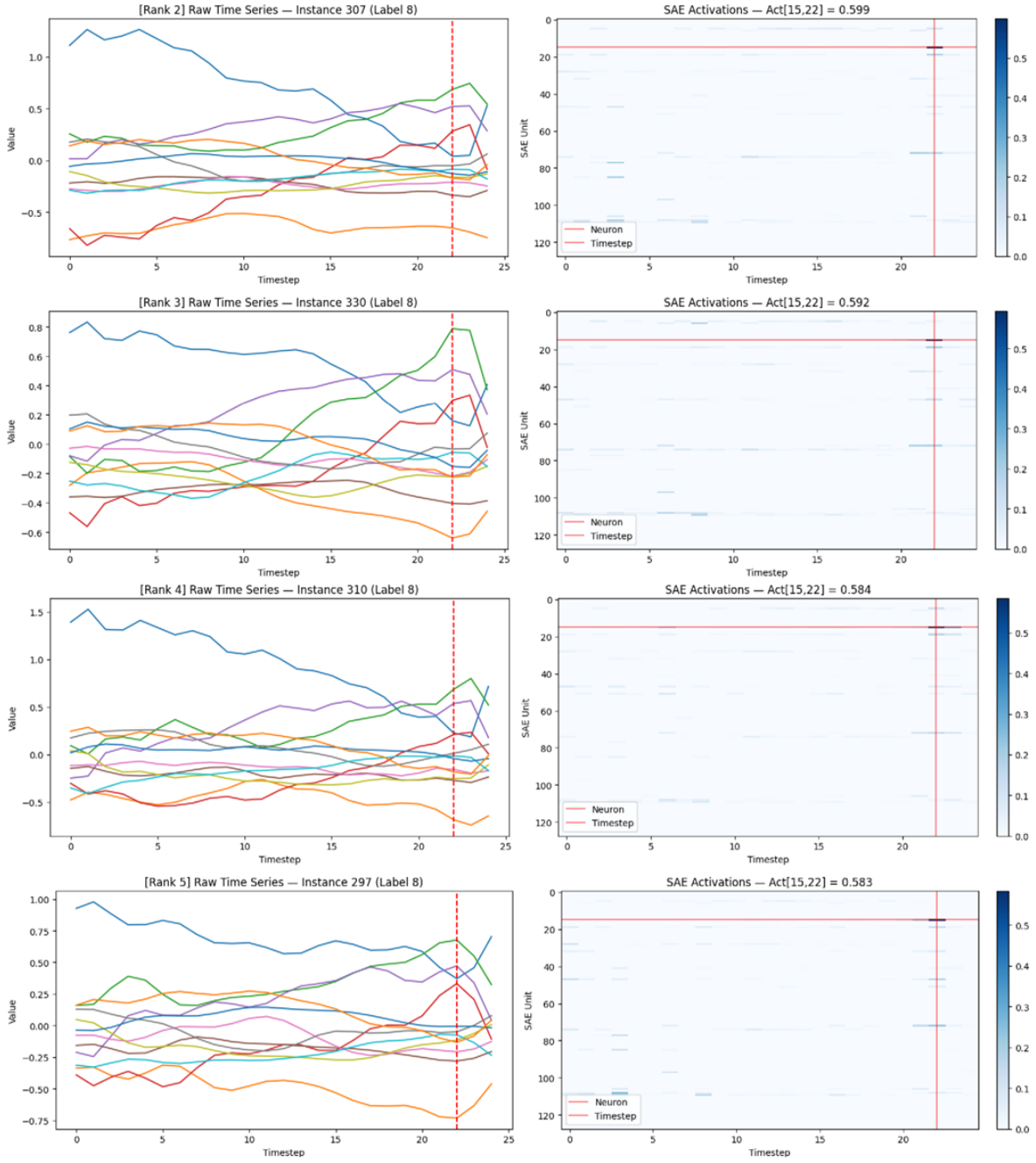


Figure 22: Top-activating test instances for SAE neuron 15 at timestep 22. These instances all belong to Class 8 and exhibit similar temporal patterns, suggesting neuron 15 encodes a class-discriminative feature.

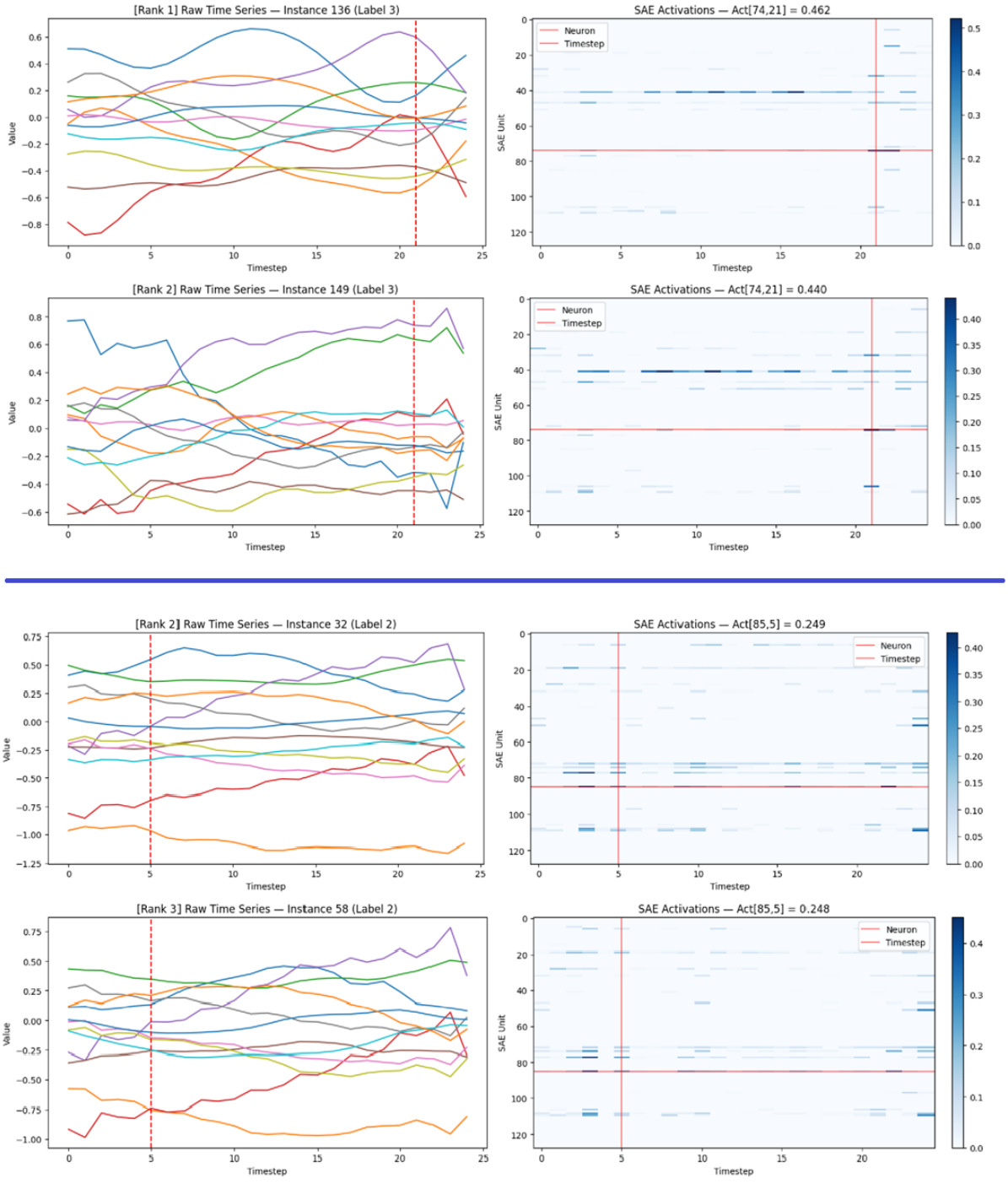


Figure 23: Comparison of sparse activations in Class 2 vs Class 3 instances. Highlights class-selective neurons such as neuron 85 and 78 (Class 2) and neuron 41 (Class 3).

List of Interesting SAE Neurons

The following list summarizes selected SAE neurons trained on activations from Encoder Block 0, Linear Layer 2 of the TST model trained on JapaneseVowels. These neurons exhibit strong, selective activation patterns associated with specific classes and temporal features. These can be inspected by running the `SAE.ipynb` notebook found in the GitHub repository.

Neuron 5, Timestep 23 — Highlights **Class 9**.
- Blue line on top, followed by purple, orange consistently at the bottom.

Neuron 5, Timestep 22 — Highlights **Class 8**.
- Green on top, followed by blue and purple (or all three close).
- Red line rises prominently.

Neuron 5, Timestep 21 — Highlights **Class 7**.
- Purple and blue dominate; orange is not lowest, unlike other classes.

Neuron 5, Timestep 20 — Ambiguous.
- Class 8/7: similar activation patterns.
- Class 1: Green on top, blue next—resembles Class 8; instance misclassified as Class 8.

Neuron 74, Timestep 22 — Strong activations for **Class 3**.

Neuron 41, various timesteps — Strong for **Class 3**.
- Weaker in low-confidence Class 3 cases.
- Notably active in Instance 44 (True: Class 2, Misclassified as Class 3); resembles Class 3 sparse pattern.

Neuron 51, Timestep 24 — Strong for **Class 9**.

Neuron 5, Timestep 23 — Reappears strongly for **Class 9**.
- Back-to-back activations with Neuron 51 suggest a temporal motif.

Neurons 108 & 109, various timesteps — Selectively active for **Class 7**.