FINDING THE ZEITGEIST IN TIME SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) achieve strong zero-shot and transfer performance across diverse tasks, yet their internal representations remain poorly understood. Although sparse autoencoders (SAEs) have emerged as a powerful tool for mechanistic interpretability in language and vision models, this has not yet been established with TSFMs. We conduct the first large-scale exploration of their internal representations using SAEs, uncovering sparse features that align with interpretable temporal patterns. Moving beyond only probing for hypothesized concepts, SAEs cast opaque model internals into disentangled, often monosemantic features. To structure their large latent space, we establish *Zeitgeist*, an unsupervised method for fully automated characterization of individual latent features. Zeitgeist enables creating a characterizing atlas of activating geometries, captions, frequency, locality, and positional affinity, describing the inner workings of TSFMs. We demonstrate Zeigeist's generality across 9 TSFMs, spanning 3 architectural families, and provide new insights into their systematic differences in reasoning. Moreover, we show that Zeitgeist's internal representations are not only a tool for interpretability but can also be leveraged for targeted interventions on a TSFM's activations, providing a strong foundation for steering downstream applications.

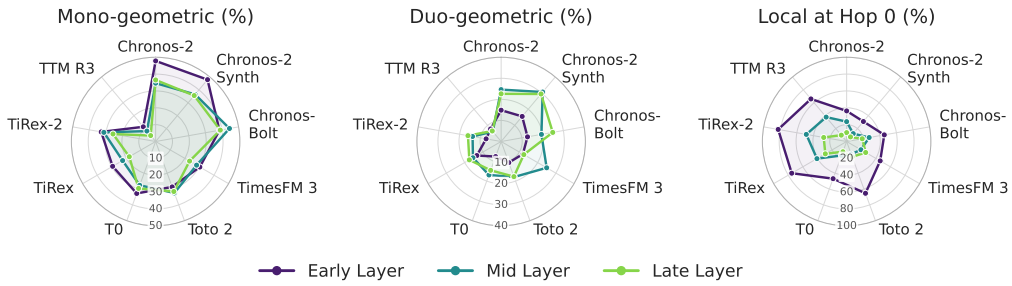


Figure 1: **Zeitgeist unveiling interpretable characteristics, by model and layer depth:** Using Zeitgeist, one obtains an explanatory set of properties of the latent features of an SAE fitted on a TSFM, such as geometricity and locality, that enable scalable introspection and downstream steering.

1 INTRODUCTION

Time series foundation models (TSFMs) have recently emerged as a powerful new paradigm for forecasting and representation learning across diverse temporal domains. By adapting architectures originally developed for language and vision data to the specific challenges of time series modeling (Ansari et al., 2024; 2025; Liu et al., 2025; Auer et al., 2025; Das et al., 2024), forecasting TSFMs have since become standard in the repertoire of many modern time series practitioners. More specifically, they are considered *foundational* due to being trained to simultaneously model a wide variety of numerical time series using a single set of weights. As a result, TSFMs achieve strong zero-shot transfer performance across many forecasting tasks (Aksu et al., 2024; Shchur et al., 2026). At present, however, the internal representations of TSFMs and how they support forecasting behavior remain poorly understood and are an active area of research (see Sec. 2).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

While linear probing has since been used to peek into TSFMs (Pandey et al., 2025; Wiliński et al., 2025; Chemeris et al., 2026), it is fundamentally limited by the diversity and completeness of the probing labels. In contrast, mechanistic interpretability (mech interp) has made substantial progress in Large Language Models (LLMs), driven by extracting hidden activations and subsequent self-supervised learning of dictionaries of their representation using sparse autoencoders (SAEs) (Cunningham et al., 2024; Bricken et al., 2023; Stevens et al., 2025). These SAEs act as lenses into the otherwise opaque inner processes of these overwhelmingly large foundation models, without requiring comprehensive a priori hypotheses of the concepts contained therein. They have proven themselves as a particularly effective tool for discovering disentangled, often monosemantic features (Härle et al., 2025) that humans can inspect. Their success has since sparked further progress by moving beyond deciphering individual features and neurons to circuits (Dunefsky et al., 2024) and to multimodal models (Shen et al., 2025; Pach et al., 2025).

These approaches have shown that seemingly complex behaviors can often be explained by relatively simple, reusable components embedded in high-dimensional residual streams, even without specifying target concepts in advance. Since modern TSFMs closely mirror LLMs in both architecture and training dynamics, a natural question arises: Do similar interpretable features exist within TSFMs, and can the same methods be applied to lead to the same level of success? This work shows how such a transfer is possible with a diverse set of both SAE and foundation model architectures. Hereby, we move substantially beyond the initial soundings of Mishra (2026), which considered only a single SAE architecture and Chronos-1, a time-step-wise TSFM. While early work investigated dictionary learning on classical per-domain models (Kalnāre et al., 2025; Yıldırım, 2026), we argue that SAEs are particularly well-suited for foundation models due to the extremely diverse nature of their training data, rendering hypothesis-driven probing isufficient for obtaining a exhaustive description of the model.

Once trained, a major challenge with SAEs is their complexity due to their vast size, which has limited prior attempts to use them for time series models (Kalnāre et al., 2025; Mishra, 2026; Yıldırım, 2026). To aid human interpretability, our proposed method *Zeitgeist* structures the SAE latent space along five complementary axes, such as geometricity and locality, yielding deep insights into a TSFM's reasoning (see Fig. 1). Namely, we combine SAEs with an automated interpretability (auto-interp) pipeline that measures the individual features' activating geometries, textual captions, frequency, locality, and positional affinity. This allows for building an atlas that describes the otherwise unwieldy hidden representations of TSFMs using human-interpretable features. Ultimately, this unlocks new applications, of which we demonstrate steering using interventions on latent activations.

Contributions. **i)** We provide, to the best of our knowledge, the first comprehensive adaptation of SAEs to TSFMs, learning sparse and interpretable features, surveying three widely used dictionary learning architectures on 9 foundation models. **ii)** This enables unsupervised and extensive exploration of TSFM internals, whereas existing probing methods relied on a predefined and limiting set of hypothesis features. **iii)** We propose *Zeitgeist*[1], an automatic interpretability method to characterize SAE latents by five dimensions tailored to time series. This enables large-scale use across models, layers, and configurations. **iv)** We show how Zeitgeist yields atlases that render thousands of latent features systematically explorable and comparable across multiple SAEs and model families. To our knowledge, this is the first large-scale systematic characterization of SAE feature spaces in TSFMs.

2 Related Work

Interpreting Temporal Representations. Probing studies examine which predefined temporal concepts are accessible in TSFM hidden states and how their encoding varies across layers (Pandey et al., 2025; Chemeris et al., 2026). Such probes test the recoverability of specified properties, rather than enumerating the features learned by the model. Wiliński et al. (2025) additionally connect representation analysis to concept-informed steering. Complementary work studies rank structure and component redundancy (Yu et al., 2026; Bao et al., 2026), or tests forecast responses to controlled temporal interventions (Jander et al., 2026). For task-specific models, RioT uses feedback on explanations to discourage reliance on spurious correlations (Kraus et al., 2025). Zeitgeist complements these targeted analyses by learning features without temporal concept labels and characterizing their activating patterns afterward.

[1]Code available at anonymous.4open.science. Example checkpoints and atlases available at Huggingface.

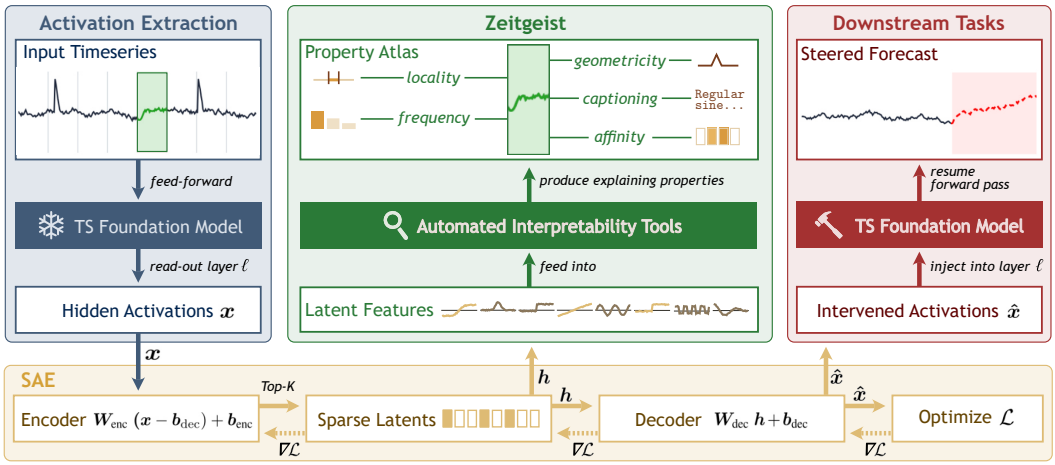


Figure 2: **Zeitgeist provides an unsupervised lens into TSFM hidden states:** We extract the residual-stream activations $\boldsymbol{x}$ after block ℓ of a frozen TSFM and train an SAE to obtain a sparse set of latent features. We make the obtained human-interpretable features explorable via an atlas of geometricity, locality, frequency, position affinity, and captions, thereby unlocking downstream applications such as steered forecasting that is grounded in a model's observed reasoning patterns.

Sparse Autoencoders for Mechanistic Interpretability. SAEs decompose hidden states into sparse combinations of learned directions, enabling feature discovery beyond individual neurons (Bricken et al., 2023; Cunningham et al., 2024). This approach has scaled to large language models (Templeton et al., 2024; Gao et al., 2024). Because large dictionaries are impractical to inspect manually, automated interpretation generates natural-language feature explanations at scale (Paulo et al., 2025). Structured variants organize features hierarchically with Matryoshka SAEs (Bussmann et al., 2025) or encourage consistency across neighboring representations with Temporal SAEs (Bhalla et al., 2026). These methods provide the SAEs and automated-interpretation foundations for Zeitgeist.

Sparse Features in Time Series Models. Kalnāre et al. (2025) combine SAEs with activation patching for task classification. In PatchTST forecasting, Yıldırım (2026) find limited impact from dictionary expansion and latent interventions, cautioning against assuming strong superposition in all temporal models. Oublal et al. (2026) explain black-box time-series models via sparse representations without access to hidden states. Closest to our work, Mishra (2026) trains TopK SAEs on Chronos-T5-Large, links features to synthetic temporal patterns, and tests them via ablations, while Lakhanpal (2026) apply SAEs to MOMENT and studies latent steering. Extending these single-model studies, Zeitgeist enables automated analysis across TSFMs and SAE variants, combining activation geometry, frequency, locality, positional affinity, and captions into an explorable, comparable atlas.

3 Zeitgeist

To peek into TSFMs, Zeitgeist first extracts activations from their hidden layers (Sec. 3.1) and employs SAEs as a lens into them Sec. 3.2. Thereafter, we show how the SAE's latent space can be mapped out for effective exploration (Sec. 3.3). Finally, we show how Zeitgeist's latent features can be leveraged in downstream applications (Sec. 3.4). Fig. 2 provides a complete overview of Zeitgeist.

3.1 Extracting Hidden States from TSFMs

We now describe how hidden states $\boldsymbol{x}$ are obtained. Let $\mathcal{M}$ be a pretrained Transformer model operating on patchified time series inputs of length T. Given a time series window $\boldsymbol{s} \in \mathcal{S} \subseteq \mathbb{R}^T$, we partition it into P non-overlapping patches of length L such that $T = P \cdot L$. After patch embedding and positional encoding, the sequence of patch tokens is processed by multiple Transformer blocks. At layer ℓ, the model produces a sequence of residual stream hidden states $X^{(\ell)}(\boldsymbol{s}) = (\boldsymbol{x}_1^{(\ell)}(\boldsymbol{s}), \ldots, \boldsymbol{x}_P^{(\ell)}(\boldsymbol{s}))$ with $\boldsymbol{x}_p^{(\ell)}(\boldsymbol{s}) \in \mathbb{R}^{d_{\text{model}}}$. Each vector $\boldsymbol{x}_p^{(\ell)}(\boldsymbol{s})$ represents the model's internal encoding of a temporal patch at position p.

For SAE training, we treat individual hidden states as independent samples and construct a dataset $\mathcal{D}^{(\ell)} = \{\boldsymbol{x}_p^{(\ell)}(s)\}_{\boldsymbol{s}\in\mathcal{S},p\in\{1,\dots,P\}}$ aggregated across time series, windows, and patch positions. Note that neighboring patches are not truly independent, yet treating them separately allows decoding and interpreting each token's specific representation (Bricken et al., 2023). We exclude any special tokens from the dataset, as is common in SAE training (Bloom et al., 2024).

3.2 Sparse Autoencoders for Temporal Dictionary Learning

We use established sparse autoencoder (SAE) architectures to decompose hidden states $\boldsymbol{x} \in \mathbb{R}^{d_{\text{model}}}$ of a frozen TSFM into sparse features. An encoder E and sparsifying nonlinearity σ produce an overcomplete code $\boldsymbol{h} \in \mathbb{R}^{d_{\text{sae}}}$, from which a linear decoder reconstructs the hidden state:

$$\boldsymbol{h} = \sigma(E(\boldsymbol{x})), \qquad \hat{\boldsymbol{x}} = D(\boldsymbol{h}) = \boldsymbol{W}_{\text{dec}}\boldsymbol{h} + \boldsymbol{b}_{\text{dec}} \ . \tag{1}$$

The decoder columns form a learned dictionary. Importantly, training uses no temporal concept labels or other supervised guidance. All variants use BatchTopK sparsity (Bussmann et al., 2024), which shares an activation budget across a batch. Matryoshka SAEs (Bussmann et al., 2025) additionally require nested dictionary prefixes to reconstruct the input. Our default, the Temporal SAE (T-SAE; Bhalla et al., 2026), combines nested reconstruction with a contrastive loss encouraging similar high-level codes for consecutive patches within the same time-series window. BatchTopK and Matryoshka serve as ablations without this temporal loss. Full formulations and training objectives are provided in Sec. A.1. The learned features are characterized after training to form the atlas in Sec. 3.3.

3.3 Mapping out the Latent Space

The learned dictionary provides latent features, but does not by itself explain which temporal patterns they represent. We build an inspectable atlas by relating feature activations to their inputs on an in-domain probing corpus. A *firing index* records firing counts and sampled activating series with their per-position activations. This supports five complementary analyses of feature usage, context dependence, and activating shapes.

Frequency. The firing rate measures how widely a feature is used across the probing corpus. For N probed patches, we compute $\rho_i = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}[h_{n,i} > 0]$, where $h_{n,i}$ is the SAE activation of feature i on patch n. We restrict subsequent analyses to features with $\rho_i \geq 10^{-6}$, focusing on those that fire at least once per million patches.

Positional Affinity. Features may favor particular context positions, such as the beginning or end of a window. We average each feature's activation magnitude at each position across its indexed firing series, then normalize the profile to sum to one. The resulting distribution describes where activation is concentrated, separately from how frequently the feature fires overall.

Locality. A feature's activation can reflect both the current patch and its surrounding context. To assess this dependence, we fix an activating patch and its r neighboring patches on each side, perturb the remaining context, and rerun the TSFM and SAE. Across activating examples, we record whether the feature remains active at the original position. A feature is *Hop-r local* when a one-sided exact binomial test supports an activation-survival probability above 0.9 at significance level 0.05. We report the smallest tested radius that passes; Hop-0 retains only the activating patch. We correct the perturbed region to preserve whole-series mean and variance, controlling changes induced by input normalization. This classification is relative to the tested perturbations . See App. A.2 for details.

Geometricity. A feature's activating patches need not share a single shape. We therefore cluster input patches within each feature to identify recurring geometries before assigning a textual description. This distinguishes features associated with one dominant shape from those associated with several different shapes. From each selected firing series, we extract the maximally activating patch, preventing repeated firings within one series from dominating the sample. We standardize each patch to compare shape rather than absolute level or amplitude. We then apply average-linkage hierarchical clustering, cutting at 80% of the height, using the minimum Euclidean distance over relative shifts of -1, 0, and $+1$ time steps, with distances rescaled for the overlap length. This groups visually similar shapes despite small temporal offsets, without allowing arbitrary warping of the time axis as in DTW. Clusters containing less than 5% of sampled patches are pooled into a residual group rather than treated as separate recurring shapes.

A feature is *mono-geometric* if its largest retained cluster accounts for at least 70% of all sampled patches, *duo-geometric* if the two largest jointly reach 70% but neither does alone, and *poly-geometric* otherwise. These categories summarize the concentration of activating shapes, while the retained clusters supply representative examples for captioning.

Captioning. We turn the clustered shapes into readable descriptions in two stages. First, an LLM captions each retained shape cluster from its *medoid*, the member patch with the smallest total within-cluster distance, using a few-shot prompt. Unlike an average trace, the medoid is an observed example and does not blur slightly shifted patterns. Second, a feature-level prompt combines the cluster captions and their shares into a title and grouped descriptions, merging similar patterns while preserving distinct ones. The residual group is not captioned. Both prompts are provided in App. A.3.

3.4 Downstream Applications: From Inspection to Intervention

Beyond describing a model's internal representations, the atlas provides an interface for investigating their role in forecasting. We use selected SAE features as intervention directions and examine how the forecast changes when these directions are added to the hidden state. This connects inspection of activating patterns to intervention in the frozen model.

For a sparse latent offset $\delta\boldsymbol{h}$, we decode a hidden-state direction and add it with a factor α, which decides the strength of the intervention on the hidden states at selected context positions $\mathcal{P}$:

$$\boldsymbol{x}'_p = \boldsymbol{x}_p + \alpha\boldsymbol{d} \qquad \text{with} \qquad p \in \mathcal{P} \qquad \text{and} \qquad \boldsymbol{d} = D(\delta\boldsymbol{h}) - D(\mathbf{0}) = \boldsymbol{W}_{\text{dec}}\delta\boldsymbol{h}\ . \tag{2}$$

The edit is applied at the output of the TSFM block associated with the SAE before resuming the remaining model computation. It retains the original hidden state and its reconstruction residual, while leaving other states unchanged at this layer. Structural and future-patch tokens are excluded from $\mathcal{P}$, and $\alpha = 0$ recovers the native computation. Neither the observed inputs nor the model parameters are modified. In Sec. 4.5, paired examples determine the latent offsets, and validation data select their support and strength.

4 Experiments

Having established Zeitgeist as a pipeline from hidden states to an atlas of interventable features, we now put each stage to the test. Our experiments answer the following questions:

- (Q1) Is unsupervised representation extraction with SAEs a viable lens into TSFMs? How does layer depth affect representations?
- (Q2) How do SAE architectures, training corpora, and hyperparameters affect extracted features?
- (Q3) What can we learn about the otherwise opaque inner workings of TSFMs and differences across architectures?
- (Q4) How do representations uncovered by Zeitgeist look like for typical time series?
- (Q5) Can the identified features be used for targeted steering of foundation model forecasts?

TSFMs. We study encoder-only transformers (Chronos-2 and Chronos-2-Synth (Ansari et al., 2025), Chronos-Bolt (Ansari et al., 2024)), decoder-only transformers (TimesFM-3.0 (Das et al., 2024), Toto-2.0 (Khwaja et al., 2026), T0 (Meyer et al., 2026)), recurrent/hybrid models (TiRex (Auer et al., 2025), TiRex-2 (Podest et al., 2026)), and an MLP-Mixer (TTM-R3 (Laglil et al., 2026)). See Tab. 4 in App. B for details. All TSFMs were kept frozen throughout. **Training and Probing Data.** This work focuses on univariate time series; therefore, we first drew data from the diverse, large-scale KernelSynth (Ansari et al., 2024), which consists of 1M univariate time series of length 1024 with no missing values. Additionally, we source data from the GIFT-Eval (Aksu et al., 2024) pretrain corpus. Specifically, we take the first 5,000 time series, possibly containing missing values, from each constituent dataset, truncated to the last 1024 time steps each. Together, this amounts to 88.9M tokens from 1.4M series. **Captioning LLM.** We use `gpt-5.6-luna` at effort `medium`. **Synthetic Shapes Dataset.** To assess whether at least some suspected key temporal structures are encoded in the hidden representations, we construct a controlled probing dataset with 14 synthetic temporal concepts. See App. C for details. **Sparse Autoencoders.** We provide extensive experiments on varying hyperparameters in Sec. 4.2 and App. D.2.

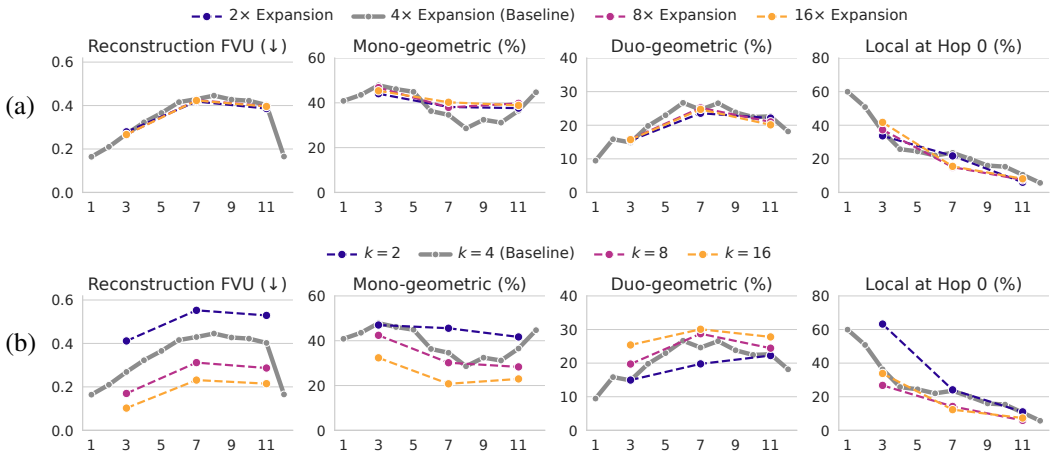


Figure 3: **Overview of SAE hyperparameter influences across layers of Chronos-2** (horizontal axis). **(a)** The expansion factor has a limited effect on the learned representations. **(b)** The sparsity parameter k strongly influences the discovered features, trading off FVU against ease of interpretability.

Table 1: **Adding the real-world GIFT-Eval data to the T-SAE training and probing corpus to KernelSynth improves mono-/duo-geometricity in all and locality in later layers.**

Chronos-2 Layer	FVU (↓)	Mono (%)	Duo (%)	Locality (%)
2 (early)	0.256 (+0.013)	44.6 (+3.2)	12.4 (+2.5)	40.0 (-3.7)
6 (mid)	0.410 (+0.020)	32.6 (+2.0)	19.3 (+5.3)	24.8 (-1.2)
10 (late)	0.424 (-0.022)	32.0 (+4.5)	14.9 (+7.7)	8.8 (+1.7)

As a common reference baseline throughout, we selected T-SAE, trained for 100k steps on the KernelSynth and the GIFT-Eval pretraining subset, with expansion factor $4\times$, $k = 4$, $\tau = 0.05$, and $\lambda_{\text{temp}} = 0.05$. Full details are provided in App. B.

4.1 Training SAEs as Lenses into TSFMs (Q1)

Before applying SAEs, we first verify that intermediate hidden states of Chronos-2 encode meaningful temporal structure. Following prior TSFM interpretability work (Pandey et al., 2025; Wiliński et al., 2025; Chemeris et al., 2026), we train linear probes directly from frozen hidden states of Chronos-2. Specifically, we fit logistic regression classifiers supervised with labels of synthetic temporal concepts (see App. C). Tab. 2 summarizes linear probe performance across all evaluated temporal concepts and compares it with the three SAE architectures. These follow self-supervised training, and the best-matching features are selected afterward. Overall, this demonstrates that the selected layers contain temporal concepts that can be extracted with self-supervised SAEs, providing a meaningful substrate for further representational analysis. Detailed per-concept precision, recall, and F1 scores for the overall best method T-SAE across layers are reported in Fig. 9 in App. D.1. Learning discrete, interpretable representations for continuous, complex time series entails some approximation error. Tab. 3 quantifies this effect by assessing forecasting performance change on the univariate GIFT-Eval test set, showing that later layers are less affected by SAE sparsity.

4.2 Effects of Differing Training Recipes (Q2)

Throughout this work, we seek combinations of data corpora, SAE architectures, and their configurations that appropriately reflect the TSFMs' internals and achieve low reconstruction error. Furthermore, to aid human interpretability, we select by high fractions of local and mono-/duo-geometric features and smaller latent spaces for a better overview. This section discusses tradeoffs encountered when training SAEs on TSFMs in practice.

Table 2: **SAEs extract known temporal concepts.** The best self-supervised method is highlighted **bold**.

Chronos-2 Layer of 12	Geometric Mean F1 (↑)			
	Superv. LP	T-SAE	B.TopK	Matry.
2 (early)	0.999	**0.777**	0.724	0.645
6 (mid)	1.000	**0.779**	0.763	0.724
10 (late)	1.000	0.644	**0.697**	0.681
Mean	0.999	**0.731**	0.727	0.683

Table 3: **Quantifying the cost of sparsification.**

Chronos-2 Layer of 12	GIFT-Eval CRPS Δ (↓)	
	Median	Geom. Mean
2 (early)	+22.67%	+25.57%
6 (mid)	+14.95%	+23.63%
10 (late)	+ 7.74%	+13.03%
Mean	-	+20.74%

Expansion Factor. Since Fig. 3a showed Zeitgeist to be very robust across many sizes of the latent space, we opted for $\times 4$ to limit its size. **Sparsity Parameter.** Selecting k has grave and monotonically systematic impacts on the latent space (see Fig. 3b). Firstly, reconstruction quality substantially improves with the added capacity of a larger k.

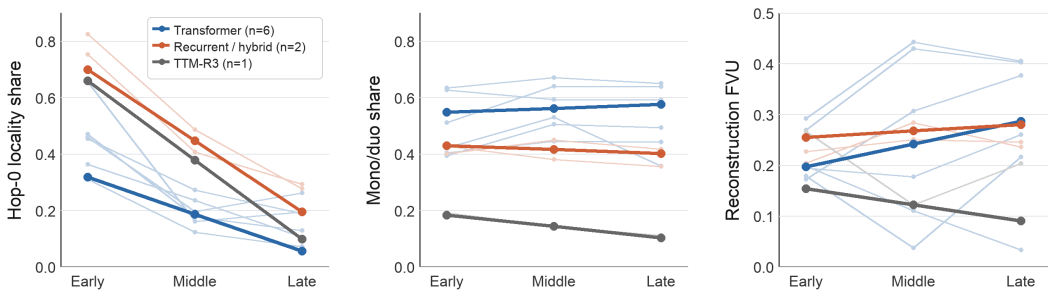
However, mono-geometricity and locality strongly benefit from increased sparsity (lower k). Overall, we find our baseline $k = 4$ to match $k = 2$ at layers three and eleven on those metrics, while offering substantially lower FVU. **Training and Probing Corpus.** Choosing the right and probing training data is key to a faithful representation of a foundation model's representations and the search for interpretable features. To ensure no distribution shift between training and probing, we employ the exact same data. We base our experiments on Kernel Synth because of its diverse range of shapes. Tab. 1 shows the effect of additionally supplying a subset of the real-world GIFT-Eval pretrain data as well: The reconstruction quality (FVU) on this more extensive domain remains similar, while substantially more features are mono-/duo-geometric. Slightly fewer features are local in earlier but more in later layers. Together with the added realism, this motivated including GIFT-Eval pretrain data as our baseline. **Further Details.** In addition, App. D.2 investigates the effect of the three SAE architectures, several different training step counts, and different temporal loss configurations.

4.3 Insights Across TSFMs (Q3)

With the SAE recipe fixed, we now use Zeitgeist to examine how representations evolve across model depth and architecture. Building on Fig. 1, we compare early, middle, and late layers of nine TSFMs using matched T-SAEs ($4\times$ expansion, $k = 4$, 100,000 training steps) and a common training and probing mix. Fig. 4 reveals a consistent depth trend: The share of Hop-0 features decreases from early to late layers in every model, suggesting increasing dependence on context beyond the activating patch. Geometricity follows a different pattern. The combined mono-/duo-geometric share remains broadly stable in Transformers, but decreases in the xLSTM-based models and TTM-R3. Thus, broader context dependence can coexist with similarly concentrated activating shapes: locality and geometricity capture distinct aspects of the learned representations. Reconstruction provides a complementary perspective. FVU is higher in late layers than in early layers for seven of the eight non-TTM models, while TTM-R3 follows a non-monotonic trajectory. Under the matched SAE budget, later activations are therefore generally reconstructed less accurately. Overall, depth consistently shifts features away from patch-local behavior, while changes in their activating geometries differ across architectures.

4.4 An Example Feature from Auto-Interp (Q4)

The preceding analyses summarize entire dictionaries. We now illustrate how Zeitgeist makes an individual feature inspectable by combining activating geometry, an automatic caption, firing frequency, locality, and positional affinity. Fig. 5 shows feature 2735 from the layer-2 T-SAE of Chronos-2, captioned "smooth downward curves and broad humps." Of 995 sampled activating patches, 77.4% belong to a cluster of smooth declines, satisfying our mono-geometric criterion; a smaller cluster of broad humps accounts for 11.8%. The figure pairs the dominant cluster's medoid with an activating electricity-meter patch, showing their similar downward shapes after standardization, grounding the caption in observed examples. Across the probing corpus, the feature fires on 0.47% of patches. Its positional profile is not confined to a single context location, and the locality test classifies it as Hop-3 local. Its activations across the example meter's context are shown in Fig. 13. This feature also connects inspection to intervention.



(a) Hop-0 locality (b) Mono/duo geometricity (c) Reconstruction FVU

Figure 4: **Feature profiles across model depth.** Thin lines show individual models; thick lines summarize families using median per-model Theil–Sen slopes. Shares use features classified by the respective analysis; lower FVU is better.

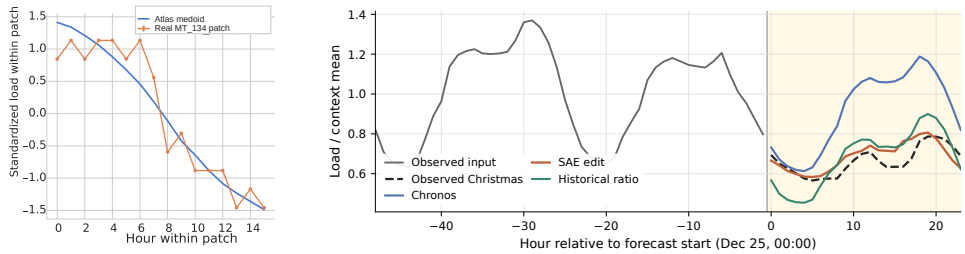


Figure 5: **Inspecting a learned feature and steering a forecast.** (left) Dominant-cluster medoid and activating patch from meter MT_134, standardized separately; the medoid comes from the probing corpus. (right) Christmas Day forecast profiles across 366 meters, normalized by each meter's context mean. The final 48 observed hours precede the shaded 24-hour forecast horizon.

A separate comparison ranked it seventh among 3,072 features by the absolute mean SAE-code difference between historical Christmas and same-weekday reference histories. It was subsequently selected for the electricity intervention in Sec. 4.5. The atlas describes patterns associated with activation; steering tests the forecast consequences of adding the corresponding decoder direction.

4.5 Targeted Steering of Forecasts (Q5)

Having made the learned features inspectable, we next ask whether they can also provide a useful interface for controlling forecasts. We test this in two complementary settings: synthetic processes with known intervention effects and a Christmas electricity forecast guided by calendar information. In both cases, we edit a small number of SAE decoder directions while keeping Chronos-2, its layer-2 T-SAE, and the observed inputs unchanged.

Controlled effects with known dynamics. The synthetic setting lets us compare a forecast change directly with the effect imposed on the underlying process. A four-state linear generator produces matched control and treated series with 512 observed steps and a 128-step forecast horizon. Eight rules apply positive or negative impulses to level, velocity, cosine, or sine states at the start of the final 16-step context patch. The paired series share process and observation noise, and the simulator supplies the known future effect Δ^* (App. E.1). For each rule, 128 training pairs estimate the mean signed SAE-code difference at the final context token. Retaining its K largest-magnitude coordinates defines the offset in Eq. (2), which we add only at that token. Another 128 pairs select $K \in \{1, 3, 5\}$ and the gain, while 256 disjoint pairs per rule are reserved for testing. At test time, the known rule selects a fixed edit of the control context; the native comparator instead receives the treated context. The experiment therefore tests rule-conditioned control, not automatic discovery of an intervention.

Effect recovery. Let $\widehat{y}_c$, $\widehat{y}_t$, and $\widehat{y}_e$ denote the median control, treated, and edited-control forecasts. We compare the native response $\widehat{y}_t - \widehat{y}_c$ and the SAE-induced response $\widehat{y}_e - \widehat{y}_c$ with Δ^*, averaging absolute errors over the horizon, pairs, and rules.

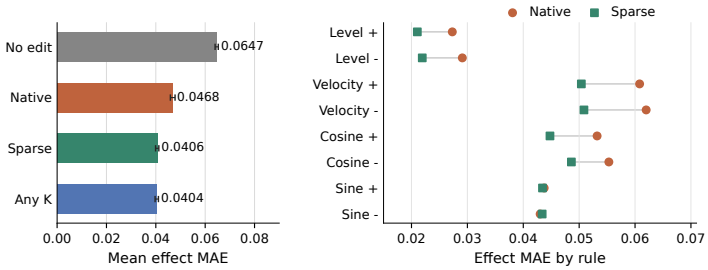


Figure 6: **Sparse steering on known dynamics.** Left: effect MAE over 2,048 test pairs with 95% bootstrap intervals; "Any K" permits larger feature supports. Right: native and sparse errors by rule.

Sparse steering reduces effect MAE from 0.0468 to 0.0406 (13.4%; Fig. 6), with a paired difference of -0.0063 and a 95% bootstrap interval of $[-0.0071, -0.0054]$. The zero-effect baseline scores 0.0647, while an expanded support search reaches 0.0404: most of the improvement is already obtained with one to five features. Improvements are strongest for the level, velocity, and cosine rules. MAE against the realized treated future also decreases from 0.1055 to 0.1025. Full profiles and selection details appear in App. E.1.

A known-date electricity forecast. Calendar events offer a setting in which the timing of an expected change is known before its effect appears in the observations. We use this information to steer a Christmas Day electricity forecast without supplying a holiday covariate to Chronos-2. Each meter's preceding 512 hourly readings, ending on Christmas Eve, provide the context for the next 24 hours. Historical Christmas/reference contrasts from 2011–12 nominate candidate features; selection on 2013 data chooses feature 2735, gain 8, and the final eight context tokens. We add its norm-rescaled decoder direction at those positions, following the construction in App. E.2. On 366 meters in 2014, mean normalized MAE decreases from 0.323 to 0.191, and the edit improves 303 meters relative to the native forecast (Fig. 5). Each meter's MAE is divided by its context mean before averaging. The historical-ratio baseline scores 0.243, compared with 0.191 for the SAE edit. Baseline construction, meter-level results, and gain sensitivity appear in App. E.2. Together, the studies show how sparse decoder edits can express known process changes and incorporate calendar information into a frozen forecaster, connecting feature inspection to targeted forecast control.

5 Conclusion

We introduce Zeitgeist, an unsupervised interpretability and steering framework based on sparse encodings of TSMFs' hidden activations, which are mapped into an inspectable atlas of temporal properties. We show that this extraction of activating geometry, captions, firing frequency, locality, and positional affinity is applicable across SAE architectures and nine TSFMs, spanning three architectural families. Across TSFMs, stronger sparsity favors local, geometrically concentrated features at the cost of reconstruction fidelity. Furthermore, patch-local features become less common at greater depth, while changes in geometricity differ across architectures. Finally, sparse decoder edits enable synthetic-effect recovery and steered forecasts relative to the native model, without changing model parameters or observed inputs.

Limitations. The atlas characterizes observed activations rather than providing a complete causal account of forecasting. Feature descriptions depend on the probing corpus; standardized patch geometries omit absolute level and amplitude, and patch-level captions can miss longer temporal dependencies. Steering evidence is limited to selected Chronos-2 experiments, including one real-world holiday event, for which overlap with model and SAE pretraining data remains unverified.

Future Work. Priorities include validating feature descriptions through targeted interventions, tracing interactions between features across layers, and extending the atlas to multivariate and multimodal representations. Connecting captions to validated interventions could also support language-guided forecast control. Zeitgeist provides a foundation for moving from inspecting temporal features toward understanding their role in forecasting.

AI USE STATEMENT

In this work, we used generative AI tools to assist in developing the software implementation of the presented method and to provide writing assistance to improve the clarity of the presentation. We have not used generative AI tools to generate, clean, or reformat any data sets; develop theoretical models or the method's conceptual frameworks; formulate, prove, or write down mathematical claims; propose or refine any hypotheses; assist with translation; or interpret scientific results. Additionally, we used generative AI tools to create and edit the code that renders the presented figures. All AI-assisted work was carefully reviewed and verified by the authors for correctness. This includes checking all generated or edited code to ensure it correctly corresponds to the method presented in the paper. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The aim of our work is to advance the frontier of interpretability in machine learning methods, and more concretely, for Time Series Foundation Models. Although enhanced interpretability is a critical ingredient for the responsible use of machine learning methods in future applications, especially foundation models trained on large corpora of data from various sources and domains, there are potential nefarious uses to consider. Namely, while revealing biases and negative patterns in foundation models may help steer their reasoning in forecasting or classification tasks, it may be exploited by adversarial entities to bias them towards malicious purposes via targeted context poisoning or adversarial perturbations. Furthermore, data protection may be an issue when models have been trained on sensitive data that could be leaked through improved interpretability methodologies, such as those presented in this work. Hence, we encourage both further research into interpretability and the use of model insights for downstream tasks, as well as the cautious use of these methods in tandem with the necessary transparency and accountability when employing Time Series Foundation Models.

REPRODUCIBILITY STATEMENT

We ensure the reproducibility of our work as follows. First, we built our experiments exclusively on open-weight models and datasets, as referenced in Sec. 4, that are freely available rather than behind limited access or a paywall. Second, we meticulously list all configuration details, such as deta mixing procedure and any hyperparameters in the corresponding sections (Sec. 4 and App. B). Third, our implementation and results are made freely available alongside this work (see anonymous.4open.science), including the code used for training, experiments on the proposed method, its ablations, and permissively usable checkpoints (Huggingface). Finally, our implementation includes a user interface that helps users of our method intuitively explore the resulting data, i.e., the extracted SAE features and their properties.

REFERENCES

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Anthony Bao, Venkata Hasith Vattikuti, Jeffrey Lai, and William Gilpin. Universal redundancies in time series foundation models. *arXiv preprint arXiv:2602.01605*, 2026.

Usha Bhalla, Alex Oesterling, Claudio Mayrink Verdun, Himabindu Lakkaraju, and Flavio P. Calmon. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability. In *International Conference on Learning Representations (ICLR)*, 2026.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. `https://github.com/decoderesearch/SAELens`, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. In *Forty-second International Conference on Machine Learning*, 2025.

Alexander Chemeris, Ming Jin, and Randall Balestriero. Aionoscope: Debugging latent-state accessibility in time-series representations. *arXiv preprint arXiv:2607.00956*, 2026.

Shein-Chung Chow, Jun Shao, Hansheng Wang, and Yuliya Lokhnygina. *Sample Size Calculations in Clinical Research*. Chapman and Hall/CRC, 3 edition, 2017.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 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.

Ruben Härle, Felix Friedrich, Manuel Brack, Björn Deiseroth, Stephan Waeldchen, Patrick Schramowski, and Kristian Kersting. Measuring and guiding monosemanticity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Mathis Jander, Wouter van Heeswijk, and Martijn Mes. Causal analysis for time series foundation models. *arXiv preprint arXiv:2608.24303*, 2026.

Matīss Kalnāre, Sofoklis Kitharidis, Thomas Bäck, and Niki van Stein. Mechanistic interpretability for transformer-based time series classification. In *International Joint Conference on Computational Intelligence (IJCCI)*, 2025.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Maurice Kraus, David Steinmann, Antonia Wüst, Andre Kokozinski, and Kristian Kersting. Right on time: Revising time series models by constraining their explanations. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2025.

Morad Laglil, Bertrand Pracca, Emilie Devijver, and Eric Gaussier. Foundation models and fine-tuning: Toward a new generation of models for time series forecasting, 2026.

Vedika Sanjeev Lakhanpal. TimeSAE: Mechanistic interpretability for time-series foundation models. In *ICLR 2026 Workshop on Time Series in the Age of Large Models (TSALM)*, 2026.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025.

Lucas Meyer, Claudio Sole, Huikan Xiang, Nicolas Li, Lucas Franceschino, Arnau Quera-Bofarull, Maarten P. Scholl, Joachim Fainberg, and Geoffrey Négiar. t_0: A time-series foundation model for forecasting with context, 2026.

Anurag Mishra. Dissecting chronos: Sparse autoencoders reveal causal feature hierarchies in time series foundation models. *arXiv preprint arXiv:2603.10071*, 2026. Accepted as a poster at the ICLR 2026 Workshop on Time Series in the Age of Large Models (TSALM).

Khalid Oublal, Quentin Bouniot, Qi Gan, Stephan Clémençon, and Zeynep Akata. TimeSAE: Sparse decoding for faithful explanations of black-box time series models. *arXiv preprint arXiv:2601.09776*, 2026.

Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *Advances in Neural Information Processing Systems*, 2025.

Atharva Pandey, Abhilash Neog, and Gautam Jajoo. On the internal semantics of time-series foundation models. In *NeurIPS Workshop on Recent Advances on Time Series Foundation Models (BERT^{2}S)*, 2025.

Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Patrick Podest, Marco Pichler, Elias Bürger, Levente Zólyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex-2: Generalizing TiRex to multivariate data and streaming, 2026.

Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting, 2026.

Shufan Shen, Junshu Sun, Qingming Huang, and Shuhui Wang. Vl-sae: Interpreting and enhancing vision-language alignment with a unified concept set. *Advances in Neural Information Processing Systems*, 2025.

Samuel Stevens, Wei-Lun Chao, Tanya Berger-Wolf, and Yu Su. Interpretable and testable vision features via sparse autoencoders. *arXiv preprint arXiv:2502.06755*, 2025.

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, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, 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.

A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2): 117–186, 1945.

Michał Wiliński, Mononito Goswami, Willa Potosnak, Nina Żukowska, and Artur Dubrawski. Exploring representations and interventions in time series foundation models. In *International Conference on Machine Learning (ICML)*, 2025.

Alper Yıldırım. Superposition is not necessary: A mechanistic interpretability analysis of transformer representations for time series forecasting. *arXiv preprint arXiv:2605.05151*, 2026.

Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Yuyang Wang. Understanding transformers for time series: Rank structure, flow-of-ranks, and compressibility. In *International Conference on Learning Representations (ICLR)*, 2026.

A METHODOLOGICAL DETAILS

A.1 SPARSE AUTOENCODERS FOR TEMPORAL DICTIONARY LEARNING

We train sparse autoencoders (SAEs) on the hidden states $\boldsymbol{x} \in \mathbb{R}^{d_{\text{model}}}$ extracted as described in Sec. 3.1, keeping the underlying TSFM frozen. A linear encoder $E(\boldsymbol{x}) = \boldsymbol{W}_{\text{enc}}\ (\boldsymbol{x} - \boldsymbol{b}_{\text{dec}}) + \boldsymbol{b}_{\text{enc}} = \boldsymbol{f}$ maps each hidden state into an overcomplete latent space, and a linear decoder $D(\boldsymbol{h}) = \boldsymbol{W}_{\text{dec}}\ \boldsymbol{h} + \boldsymbol{b}_{\text{dec}} = \hat{\boldsymbol{x}}$ reconstructs it from the resulting sparse representation:

$$\text{SAE}(\boldsymbol{x}) = D(\sigma(E(\boldsymbol{x}))) \ , \tag{3}$$

where $\boldsymbol{W}_{\text{enc}} \in \mathbb{R}^{d_{\text{sae}} \times d_{\text{model}}}$, $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{sae}}}$, and $\boldsymbol{h}$ is obtained by the non-linearity σ. The columns of $\boldsymbol{W}_{\text{dec}}$ form a learned dictionary, expressing each reconstruction as a sparse linear combination of dictionary vectors. The saved SAE is normalized so that each decoder column $\boldsymbol{W}_{\text{dec}}[:, j]$ has unit norm. Importantly, we train without supervision from temporal concepts, preventing bias from specifying incomplete concept sets, as in linear probing.

Batch-level sparsity. Following Bussmann et al. (2024), we use BatchTopK throughout to enforce sparsity across a batch rather than independently for each hidden state. For a batch of N samples, let $\boldsymbol{F} = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N]$. The sparse activation matrix is $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N] = \text{BatchTopK}_k(\boldsymbol{F}) = \sigma(\boldsymbol{F})$. The operator retains the Nk largest entries across the entire matrix and sets the remaining entries to zero. Individual samples can therefore activate more or fewer than k features while sharing a batch-level sparsity budget. At inference, the batch-dependent selection is replaced by a calibrated activation threshold θ, allowing each hidden state to be encoded independently of other samples.

Nested dictionary learning. Sparse reconstruction does not explicitly preserve features at different levels of abstraction. Matryoshka SAEs (Bussmann et al., 2025) address this by requiring nested prefixes of a shared dictionary to reconstruct the input. Let $\mathcal{M} = \{m_1, \ldots, m_J\}$, with $m_1 < \cdots < m_J = d_{\text{sae}}$, denote the prefix sizes. We define

$$\mathcal{L}_{\text{recon}}(\mathcal{M}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{m \in \mathcal{M}} \left\| \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i^{(m)} \right\|_2^2 \quad \text{with } \hat{\boldsymbol{x}}_i^{(m)} = \boldsymbol{W}_{\text{dec}}[:, 1\!:\!m]\, \boldsymbol{h}_i[1\!:\!m] + \boldsymbol{b}_{\text{dec}} \ . \tag{4}$$

The sparse code is computed once using the full dictionary and reused for all nested reconstructions. Smaller prefixes must reconstruct the input without relying on later features, encouraging early features to capture broadly applicable structure while allowing later features to specialize.

Temporal dictionary learning. Following Temporal Sparse Autoencoders (T-SAEs; Bhalla et al. (2026)), we extend the reconstruction objective in Eq. 4 with temporal consistency, using $\mathcal{M} = \{r, d_{\text{sae}}\}$ with $1 \leq r < d_{\text{sae}}$. We denote the high-level prefix by $\boldsymbol{h}_i^{\text{H}} = \boldsymbol{h}_i[1\!:\!r]$ and refer to the remaining features as low-level features $\boldsymbol{h}_i^{\text{L}}$. We form B pairs from consecutive patches $(\boldsymbol{x}_i, \boldsymbol{x}_i^-)$ within the same time-series window, so that $N = 2B$. Using paired high-level codes $\boldsymbol{h}_i^{\text{H}}$ and $\boldsymbol{h}_i^{\text{H},-}$, we define the symmetric contrastive loss and their cross-pair similarities with temperature $\tau > 0$ as

$$\mathcal{L}_{\text{temp}} = -\frac{1}{N} \sum_{i=1}^{B} \left[\log \frac{\exp(C_{ii})}{\sum_{j=1}^{B} \exp(C_{ij})} + \log \frac{\exp(C_{ii})}{\sum_{j=1}^{B} \exp(C_{ji})} \right] \text{with } C_{ij} = \frac{(\boldsymbol{h}_i^{\text{H}})^\top \boldsymbol{h}_j^{\text{H},-}}{\tau} \ . \tag{5}$$

This loss favors similarity within matched temporal pairs relative to other pairings in the batch. It acts only on the high-level codes; low-level features receive no direct temporal-consistency penalty. Our motivation is to encourage features that capture structure shared across patches while retaining capacity for patch-specific variation.

Training objectives and comparison variants. All variants optimize the joint training loss

$$\mathcal{L} = \mathcal{L}_{\text{recon}}(\mathcal{M}) + \lambda_{\text{temp}} \mathcal{L}_{\text{temp}} + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}, \tag{6}$$

where $\mathcal{L}_{\text{aux}}$ trains dead features to reconstruct full-dictionary residuals and is averaged over all N states (Bussmann et al., 2024). BatchTopK uses $\mathcal{M} = \{d_{\text{sae}}\}$; both BatchTopK and Matryoshka omit the temporal term ($\lambda_{\text{temp}} = 0$). We use T-SAE as the main method and the others as ablations, with training settings, including τ, specified in the experimental setup. The learned features are characterized after training to form the atlas in Sec. 3.3.

A.2 Robust Identification of Locality in Concepts

To assess whether a neuron $j \in \{1, \ldots, d_{\text{sae}}\}$ is global or local, one approach is to perturb the neighborhoods of the tokens where our neuron of interest fires and observe whether it continues or ceases to activate. Here, *firing* means that $E(\boldsymbol{x}^{(k)})_j > 0$, i.e., that it had to be selected by the top-k operation of the particular SAE.

In practice, we sample K random series $\boldsymbol{x}^{(k)}$ for which the concept is active.[2] To obtain i.i.d. samples, they must be drawn uniformly with replacement from the subset of all series that activate the neuron. For each $\boldsymbol{x}^{(k)}$, we independently sample a perturbation

$$\pi^{(k)} \sim \mathcal{P}, \qquad \pi^{(k)} : \mathbb{R}^T \to \mathbb{R}^T$$

from some fixed distribution of pertubations $\mathcal{P}$. Note that the inferred locality property is, therefore, relative to the chosen $\mathcal{P}$. Any perturbation $\pi^{(k)} \sim \mathcal{P}$ needs to ensure that the time steps of the focal patch where neuron j activates remain unchanged. Large sample sizes K would incur substantial computational effort, so next we need to identify a sufficiently large K so that we can be certain that the neuron under investigation is local or global.

Abstractly, we generate K samples of a binary variable $Y^{(k)} \in \{0, 1\}$ that is 1 if and only if a neuron that fired for $x^{(k)}$ still fires for $\pi^{(k)}(x^{(k)})$. Let $p = \Pr(Y^{(k)} = 1)$ given a neuron j, activating data, and distribution $\mathcal{P}$. A neuron is then deemed local if $p > \rho$, where $\rho = 0.9$ is a threshold on the fraction of samples that still need to activate j despite perturbation of its neighbors. Since p is unknown in practice, we base this decision on a finite number of observed samples. However, we still need to determine how many activating time series we need to find, perturb, and check to reliably categorize neuron j.

A.2.1 Experiment Design for Locality Tests

To this end, we need to precisely specify the hypothesis we are testing. One key goal of identifying local neurons is to enable a much simpler analysis of them. As a consequence, we should conservatively identify borderline neurons as global rather than local. Therefore, we define the null hypothesis H_0 as j being global, and the alternative H_1 that j is local:

$$H_0 : p \leq \rho \qquad \text{vs.} \qquad H_1 : p > \rho.$$

As we repeatedly sample the binary random variable Y, we perform a one-sided exact binomial test at a specified significance level $\alpha = 0.05$. Concretely, for a given sample size K, we determine a critical value

$$r(K) := \arg\min_{r' \in \mathbb{Z}} \Pr(\text{Bin}(K, \rho) \geq r') \leq \alpha,$$

allowing to reject H_0 (concluding that j is local) exactly if $\sum_{k=1}^{K} Y^{(k)} > r(K)$ (Chow et al., 2017, Chap. 5.1). By construction, this has a probability of falsely accepting H_1 (labeling a global neuron as local), which we strongly seek to avoid, of at most α.

To derive the required sample size K, we also need to specify the test's power. Specifically, we require that a neuron with true $p \geq \rho_1 = 0.95$ is detected with probability of at least $1 - \beta = 0.8$. We can now finally compute K given ρ, α, ρ_1, and β as

$$K = \arg\min_{K' \in \mathbb{N}} \Pr(\text{Bin}(K', \rho_1) < r(K')) \leq \beta = 191.$$

A.2.2 Appropriate Perturbations to Measure Locality

The following setup was used to generate the perturbed time series used to assess the locality of a feature:

$$\tilde{x}_i = x_i \cdot \big(1 + w_i \, m_i\big) + w_i \, a_i,$$

Here, x denotes the original time series, m represents the multiplicative curve (a sine), and a denotes the additive curve (smoothed noise plus white noise). The weight $w_i \in [0, 1]$ determines the degree

[2] In the future, sequential testing algorithms could be used to further reduce computational effort (Wald, 1945).

to which sample i is perturbed, depending on its distance to the target patch or patches. This weight is defined by a tanh ramp:

$$w_i = \tanh\left(\frac{d_i}{T}\right),$$

In this context, d_i denotes the distance from index i to the nearest target patch, with $d_i = 0$ for indices within a patch. Consequently, the weight is zero within the retained patches, ensuring they remain unchanged, and increases smoothly toward one as the distance from the patch increases. The temperature parameter T controls the steepness of this transition; a smaller T results in a more rapid shift from the unperturbed patch to the fully modulated series. Fig. 7 shows some example time series modulations.

Alternative methods for perturbing the time series were also evaluated; however, these approaches produced unrealistic time series. For instance, replacing relevant patches with those from another time series introduced abrupt and uncharacteristic transitions. To maintain the perturbed time series within the domain of the original series and the sampling dataset, these alternatives were excluded.

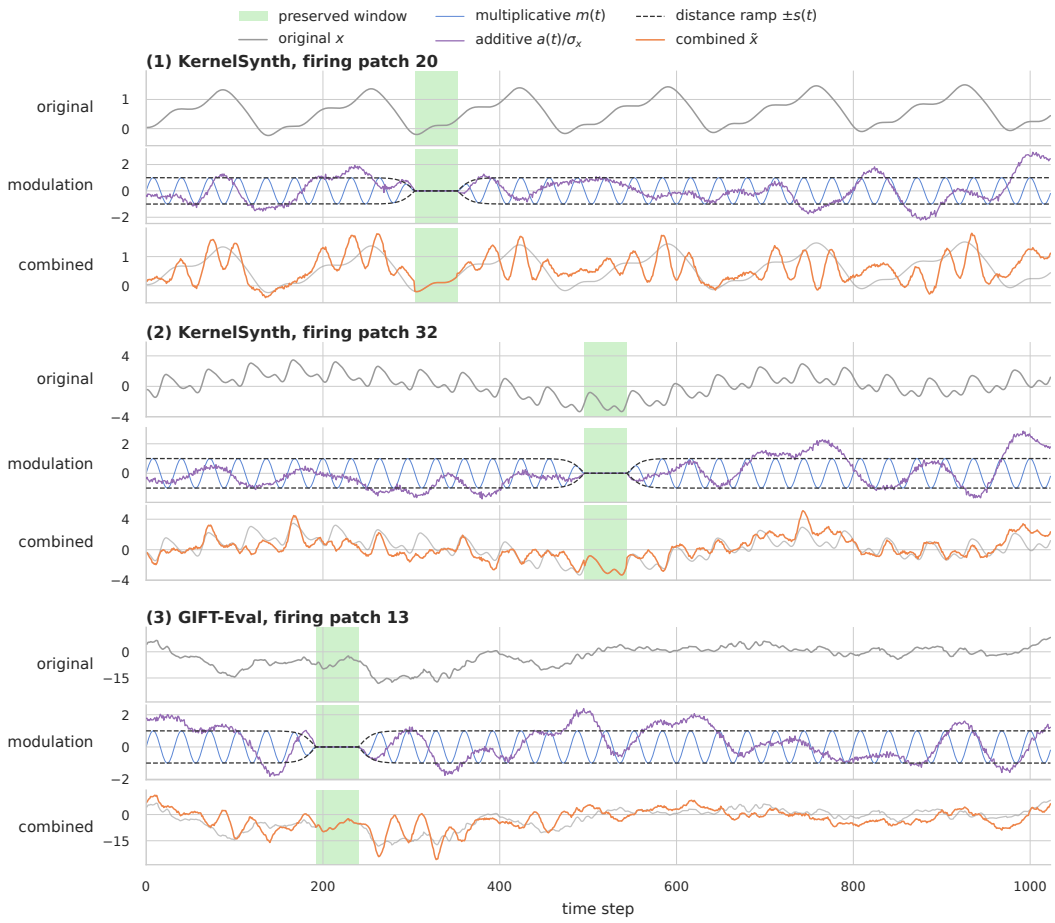


Figure 7: Examplary modulation for locality test on Time Series from KernelSynth and Gift-Eval.

A.3 Prompts Used in Captioning

We used the following two prompts for the SAE feature captioning described in Sec. 3.3 (here, for patch size 16).

Cluster System Prompt

You are a captioning system for short numeric time series (exactly 16 points, given as a Python-style list of floats, index 0 first, evenly spaced).
Write ONE short phrase (roughly 5-15 words, lowercase, no leading "the time series"/"a time series", no trailing period) that captures the single most salient pattern in the series. Mention, whenever relevant:
- the overall shape (e.g. sine, ramp, plateau, slope, spike, zig-zag, step, alternating values) - smoothness vs. noisiness (e.g. "smooth", "jagged", "high-frequency") - approximate start/end or min/max values, rounded to the nearest integer or half-integer (e.g. "from about 2 to -1") unless very flat - WHERE a change happens, as a fraction of the window or a step index (e.g. "after two thirds", "until step 10", "in the center half") - for periodic/oscillating series, the approximate cycle length in steps and the value range (e.g. "cycle-length 4 and range 1 to -2")
Output only the phrase itself — no preamble, no explanation, no quotes.
Examples of the target style:
- a smooth single-cycle sine
- high-frequency alternating values, with lower first and higher last value
- smooth, gentle downward curve from about 2 to -1
- smooth downward slope from 2 to -1 after two thirds, then upward to 1
- ramp from -2 to 2, with a plateau in the center half
- flat below zero until step 10, then rise to 2
- zig-zag with cycle-length 4 and range 1 to -2
- downward trend from 2 to 0, bottoming near the end then rebounding slightly
- smooth broad hump, rising from about -1 to reach 1.5

Summarizing System Prompt

You summarize what one neuron of a sparse autoencoder over a time series model detects. You are given the shape clusters of the input patches that activate it: one line per cluster, with its label, its share of the neuron's activations, and a short phrase describing that cluster's shape.
Produce two things.
1. A TITLE naming what this neuron actually detects — a short noun phrase (roughly 3-10 words, lowercase, no trailing period). It must be CONCRETE. Vague catch-alls are forbidden: never answer "diverse", "various shapes", "mixed patterns", "multiple patterns" or anything of that kind. If the clusters genuinely disagree, name the two or three things instead, e.g. "single-cycle sines and late downward spikes".

2. GROUPS: the same clusters, merged where merging helps. Merge two or more clusters into one group ONLY when the merged phrase is easier for a human to read than the individual lines — typically when the clusters describe the same shape and differ in one attribute you can generalize into a range. Worked examples of good merges:
- "smooth downward slope from 2 to -1" + "smooth downward slope from 1.5 to -0.5" -> "smooth downward slopes, dropping about 2 to 3 units"
- "sharp spike at step 11" + "sharp spike at step 13" -> "a single sharp spike in the last third"
- "zig-zag with cycle-length 2" + "zig-zag with cycle-length 4" -> "high-frequency zig-zag, cycle length 2 to 4"
- "flat until step 10, then rise" + "flat until step 12, then rise" -> "flat, then a rise after roughly two thirds"
- "a smooth single-cycle sine" + "a single-cycle sine starting at its peak" -> "one full sine cycle, any phase"
Often no merge is possible, and that is fine — then each cluster stays its own group, keeping its phrase. Example of a pair that must NOT be merged: "a smooth single-cycle sine" + "flat, then a sharp downward spike at the end".
Rules for the groups: use the cluster labels EXACTLY as given, list every given cluster in exactly one group, order groups by descending percentage, and set each "percentage" to the sum of its members' shares, rounded to a whole number.

B EXPERIMENTAL DETAILS

This appendix provides implementation details for the experiments in Sec. 4.

Table 4: Investigated TSFMs. $T_{\max}$ is the maximum context length supported as input.

Family	Name	Reference	L	$T_{\max}$	d_{model}	Layers	Early	Mid	Late
Transformer (enc.)	Chronos-2	Ansari et al. (2025)	16	8192	768	12	2	6	10
	Chronos-2-Synth	Ansari et al. (2025)	16	8192	768	12	2	6	10
	Chronos-Bolt (Base)	Ansari et al. (2024)	16	2048	768	12	2	6	10
Transformer (dec.)	TimesFM-3.0	Das et al. (2024)	32	15360	1280	20	2	10	18
	Toto-2.0 (2.5B)	Khwaja et al. (2026)	32	4096	2048	48	2	24	46
	T0 (beta)	Meyer et al. (2026)	32	8192	1024	24	1	12	22
Recurrent/Hybrid	TiRex	Auer et al. (2025)	32	2048	512	12	2	6	10
	TiRex-2	Podest et al. (2026)	32	2048	512	12	2	6	10
MLP-Mixer	TTM-R3 (2048-96)	Laglil et al. (2026)	64	2048	256	3	0	1	2

B.1 COMPUTING RESOURCES

All results were obtained on an *Intel Xeon Platinum 8174* CPU with 96 cores, 1.5 TiB memory, and required less than 10 TiB of disk storage. All experiments involving GPUs were run on *Nvidia Tesla V100* accelerators with 32GB of VRAM each. Activations were stored in FP16 while forward passes for all models, but the Chronos family, were performed in FP32 for numerical stability.

B.2 CONFIGURATION OF LINEAR PROBES

For the linear probing experiments, we used the logistic regression implementation of *scikit-learn*, version 1.7.2 with the `liblinear` solver, regularization strength $C = 1.0$, balanced class weights, and learning the intercept term. Optimization is run with a maximum of 1000 iterations and convergence tolerance 1×10^{-4}. Each temporal concept is treated as a separate binary classification task. Across all classifiers, we use a shared one-vs-rest, token-level evaluation per concept. The positive pool contains all tokens that represent it, while the negative pool combines $B = 8192$ from activating patches from other concepts. We train on 80% of the data and test on the remaining 20%.

B.3 CONFIGURATION OF SPARSE AUTOENCODERS

We ran a simple grid search over the hyperparameters shown in Tab. 5. Beyond the ablated settings above and the baseline described in Sec. 4, we kept the training setup fixed: We trained all SAEs on a batch size of 4096 tokens using Adam ($\beta_1 = 0.0,\ \beta_2 = 0.999$) (Kingma & Ba, 2015). We used a constant learning-rate scheduler (with configured warmup/decay horizons of 805/3222 steps and $\eta_{\text{end}} = \eta/10$), top-k threshold learning rate 0.01, decoder init norm 0.1, and b_{dec} initialization statistics computed from 50×10^3 tokens when using mean/geometric-median init. Dead-feature handling used windows of 1000 (dead-feature) and 2000 (feature-sampling) steps, and training was run in FP32. We used the *SAE Lens* library[3] (Bloom et al., 2024) implemented in *PyTorch*[4] (Ansel et al., 2024). Sparse reconstruction does not explicitly preserve features at different levels of abstraction. We do not shuffle the training tokens to implicitly group similar representations. Bin sizes used are $[384, 768, 1536, 3072]$ for Matryoshka SAEs and $[614, 3072]$ for Temporal SAEs.

Table 5: Hyperparameter grid used in SAE training. The chosen combination is underlined.

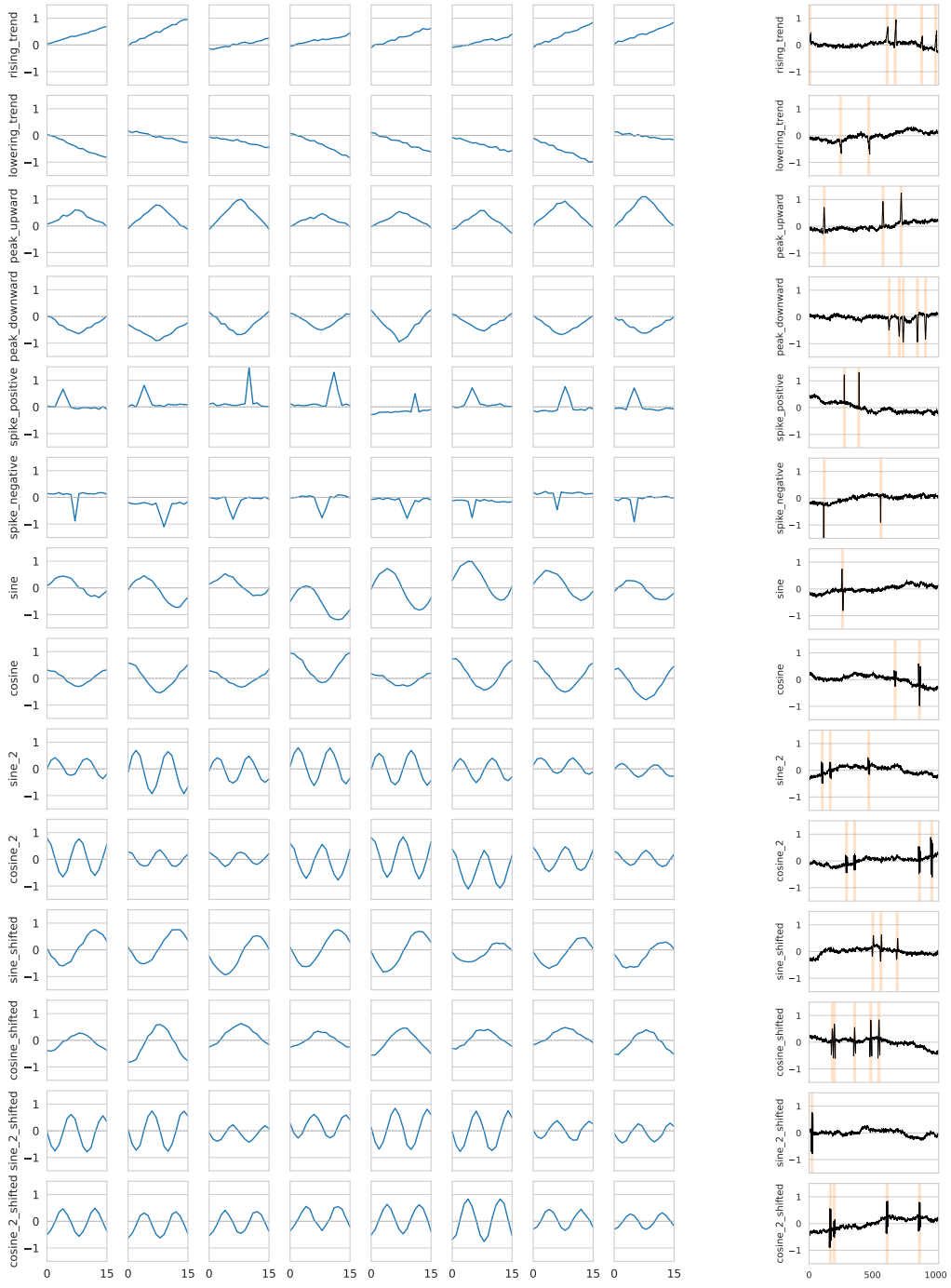
Hyperparameter	Explored Values
Decoder bias init	{zero, mean, <u>geometric median</u>}
Activation normalization	{<u>none</u>, expected average only in}
Decoder normalization	{<u>enabled</u>, disabled}
Learning rate η	$\{2 \times 10^{-5}, 5 \times 10^{-5}, 8 \times 10^{-5}, \underline{1 \times 10^{-4}}\}$
Loss weight λ_{aux}	$\{0.5, \underline{1.0}\}$

[3] `https://github.com/decoderesearch/SAELens`, version 6.44.2

[4] `https://pytorch.org`, version 2.10.0

C SYNTHETIC SHAPES DATASET VISUALIZATION

Fig. 8 visualizes samples from the synthetic dataset of known shapes. For each concept, we generate 512 time series of a slow random-walk background with additive white noise. In each series, 1 to 6 non-overlapping patches matching the tokenization boundaries are replaced with the concept pattern.



(a) Some samples probing patches. (b) Examplary time series.

Figure 8: **The synthetic shapes dataset allows simple probing for representations of 14 simple shapes.** This shows probing patches (a) and their embedding in the full time series (b), shaded orange.

D EXTENDED RESULTS

D.1 FINDING HYPOTHESIZED SHAPES

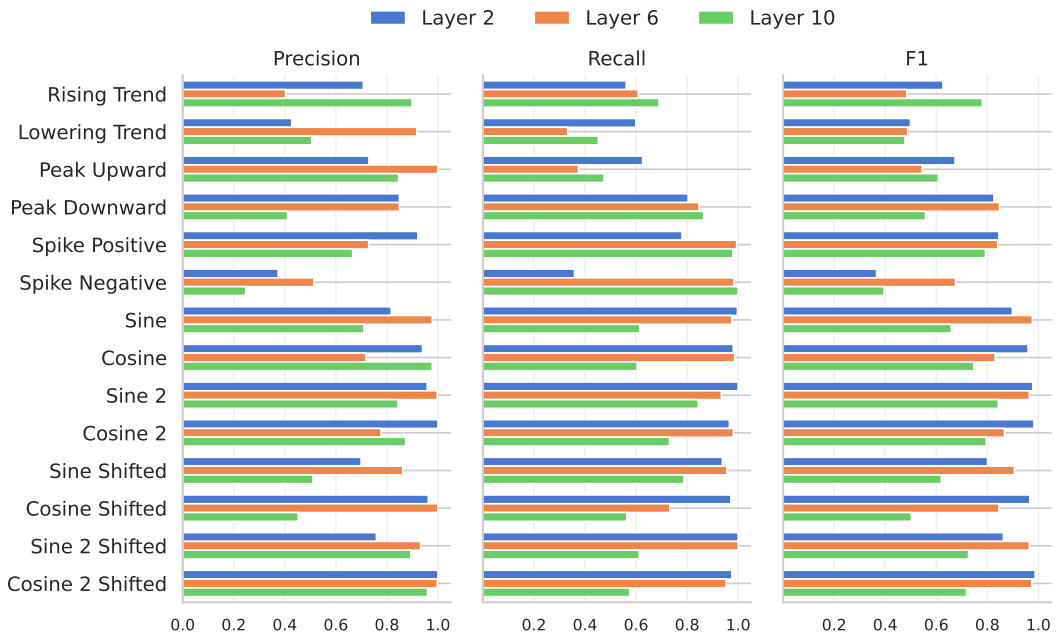


Figure 9: **SAEs can extract the hypothesized common time series concepts from the synthetic dataset.** Corresponding to the right column of Tab. 2, this shows results from T-SAE on Chronos-2 across early, mid, and late layers.

D.2 SAE HYPERPARAMETER VISUALIZATION

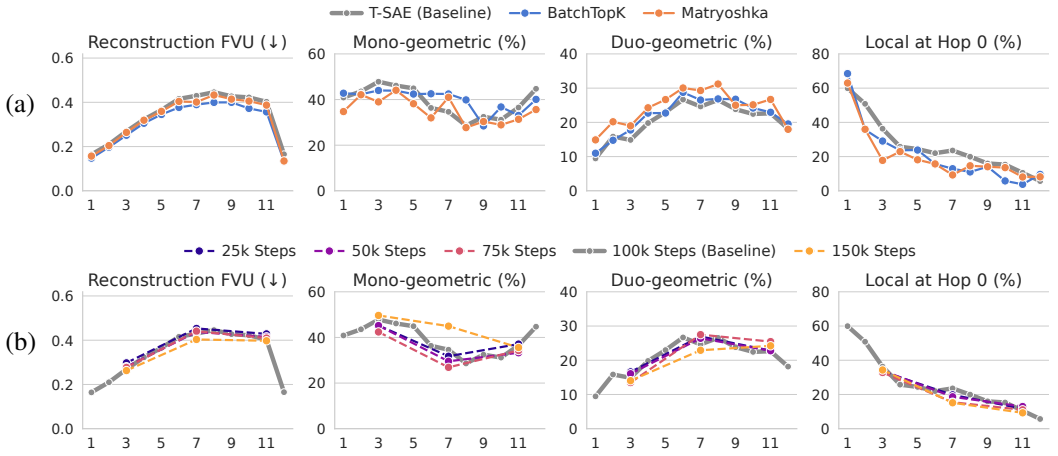


Figure 10: **(a)** T-SAE offers a good trade-off with good reconstruction quality, and many mono-geometric and hop-0 local features across layers. **(b)** 100k training steps (∼4.4 epochs) offers a good compromise, after which returns diminish.

SAE Architecture. We start by comparing the three different architectures introduced in Sec. 3.2. Fig. 10a motivated our selection of T-SAE.

Training Steps. Choosing a sufficient number of training steps while limiting compute effort is key for large-scale experiments like this. Fig. 10b shows variations of training steps, where data mixture ratios between runs have been kept constant by sub- or super-sampling. We chose 100k for all experiments throughout this work.

Temporal Dictionary Learning Coefficients Fig. 11 motivates our selection of hyperparameters: First, choosing by the reconstruction quality in the first plot, we fix the temperature to $\tau = 0.05$. Second, we see a limited effect of the temporal contrastive loss weight on mono-/duo-geometricity, while locality is best at our selected $\lambda_{\text{temp}} = 0.05$.

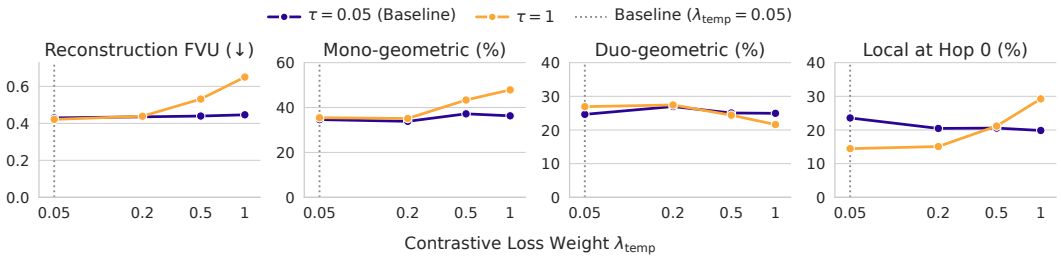


Figure 11: **Configuration of the coefficients specific to T-SAE.** Evaluated on Layer 7 (out of 12) of Chronos-2.

E Additional Steering Evidence

This appendix expands the two studies in Sec. 4.5 with their data construction, selection procedures, and complete results. Both use Chronos-2 and its frozen 3,072-feature T-SAE ($4\times$ expansion, training $k = 4$), with edits applied at the output of encoder block 2 (zero-indexed).

E.1 Synthetic Dynamics and Selection

The synthetic pairs differ only by a known impulse to one latent state. This provides an exact reference effect for evaluating the forecast changes induced by sparse edits.

Generator. The latent state $\boldsymbol{q}_t$ contains level, velocity, cosine, and sine coordinates, with transition and observation equations

$$\boldsymbol{q}_t = A\boldsymbol{q}_{t-1} + \boldsymbol{\epsilon}_t, \qquad y_t = \boldsymbol{c}^\top \boldsymbol{q}_t + \eta_t, \qquad \boldsymbol{c} = (1,0,1,0)^\top,$$
$$A = \begin{pmatrix} 0.985 & 1 & 0 & 0 \\ 0 & 0.965 & 0 & 0 \\ 0 & 0 & 0.995\cos\theta & -0.995\sin\theta \\ 0 & 0 & 0.995\sin\theta & 0.995\cos\theta \end{pmatrix}, \qquad \theta = 2\pi/32. \tag{7}$$

The initial coordinates are independent zero-mean Gaussians with standard deviations $(0.3, 0.002, 0.15, 0.15)$, followed by 128 burn-in steps. Process noise has independent Gaussian coordinates with standard deviations $(0.015, 0.0005, 0.006, 0.006)$; observation noise is $\eta_t \sim \mathcal{N}(0, 0.02^2)$.

Each pair shares its initial state and all noise realizations. At index 496, the treated state receives a single impulse $\boldsymbol{v}_i = a_i \boldsymbol{b}_r$, where $\boldsymbol{b}_r \in \{\pm\boldsymbol{e}_1, \pm 0.09\boldsymbol{e}_2, \pm\boldsymbol{e}_3, \pm\boldsymbol{e}_4\}$ and $a_i = \max(\mathrm{sd}(y^c_{i,0:511}), 0.05)u_i$ with $u_i \sim \mathcal{U}(0.9, 1.1)$. The context covers indices 0–511 and the forecast covers 512–639. The target effect is

$$\Delta^*_{i,t} = \boldsymbol{c}^\top A^{t-496}\boldsymbol{v}_i, \qquad t = 512, \ldots, 639. \tag{8}$$

It is the difference between future means conditioned on the two latent states at index 511. Linearity and shared additive noise also make it equal to the realized paired difference, up to numerical precision. Errors use the generator's original units.

Intervention and Selection. The generator draws a new trajectory for each pair (seed 260928). Each rule has 128 direction-training, 128 validation, and 256 test pairs; the optimizer uses seed 260930. Training estimates the signed mean code difference at context token 31. We retain its K largest absolute coordinates and apply the decoded offset only at that token. The inference SAE uses JumpReLU and no additional activation normalization; Chronos-2 retains its own input normalization and forecast rescaling.

The expanded search uses

$$K \in \{1, 3, 5, 10, 20, 50, 3072\},$$
$$\alpha \in \{-4, -2, -1, 0, 0.5, 1, 2, 3, 4, 6, 8, 12\}.$$

Gains are either constant or scaled by the control-context standard deviation relative to its training mean. This gives 168 candidates per rule, or 72 when $K \in \{1, 3, 5\}$. Mean validation effect MAE selects the first minimizer. All selected sparse gains are constant across test pairs (Tab. 6). The expanded search selects $(K, \alpha) = (10, 2)$ for Level + and Cosine −, and the same settings as the sparse search for the other six rules.

Evaluation. We use the median of the nine forecast quantiles $0.1, \ldots, 0.9$. Effect MAE averages absolute errors over 128 steps, then pairs and rules. Confidence intervals use 2,000 pair-bootstrap replicates within each rule (seed 260928), with equal rule weighting. Forecast-level errors are reported separately in Tab. 7; full response profiles appear in Fig. 12. The no-edit comparator predicts zero effect, not zero observations. Forecast-level errors assess the treated future rather than only its change.

Table 6: **Selected synthetic interventions.** Feature IDs are ordered by absolute training mean code difference; errors are test effect MAE.

Rule	K	α	Feature IDs	Native	SAE
Level +	5	2	2075, 631, 525, 19, 462	0.0273	0.0210
Level −	3	4	525, 117, 2075	0.0291	0.0219
Velocity +	3	12	631, 2075, 462	0.0608	0.0504
Velocity −	5	12	2075, 525, 631, 117, 1894	0.0620	0.0509
Cosine +	1	8	20	0.0532	0.0448
Cosine −	1	6	567	0.0553	0.0486
Sine +	1	12	433	0.0437	0.0434
Sine −	1	12	76	0.0430	0.0434

Table 7: **Synthetic evaluation targets.** MAE averaged equally over eight rules.

Target	Native	1–5 features	Expanded K
Known effect	0.0468	0.0406	0.0404
Treated latent-state conditional mean	0.0698	0.0651	0.0648
Realized treated future	0.1055	0.1025	0.1023

Response Profiles. Fig. 12 complements the aggregate MAE results by showing the mean forecast responses for all eight intervention rules. Sparse edits track the known effects more closely than the native forecast changes, especially for the level, velocity, and cosine rules. For the seasonal rules, the edits still move the forecast in the intended direction, but the response remains attenuated relative to the reference effect.

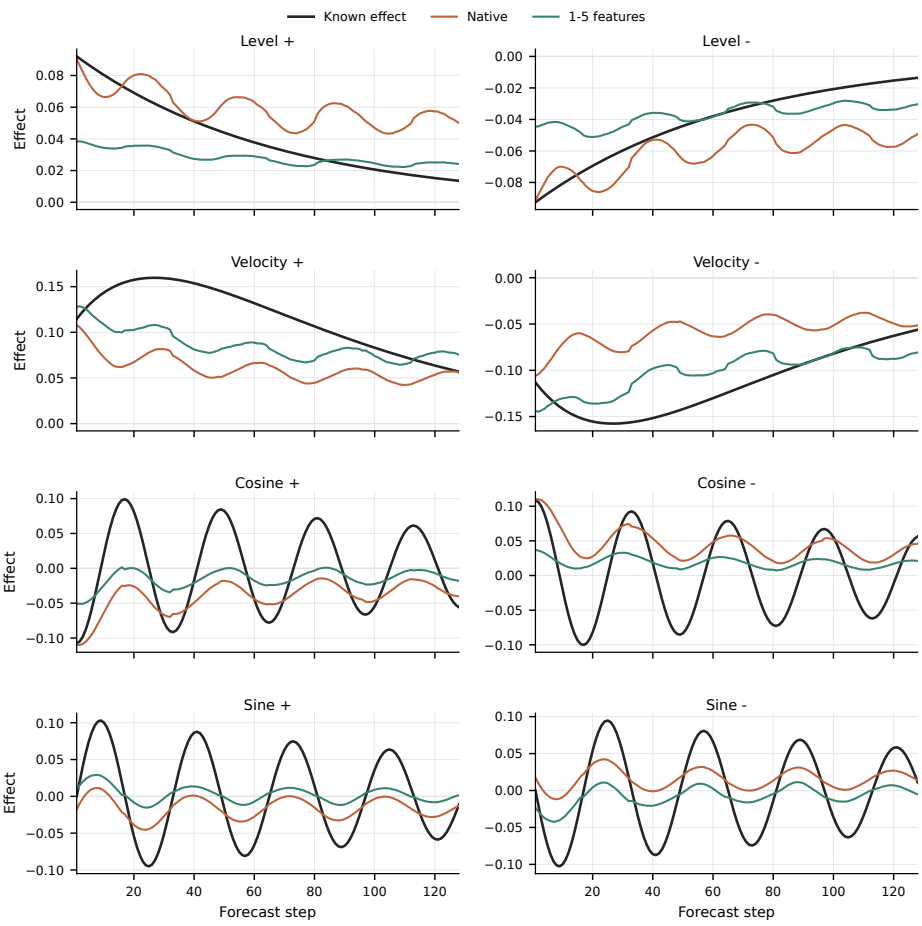


Figure 12: **Forecast responses across all eight synthetic rules.** Mean known effects and corresponding native and sparse-edit forecast changes over 256 test pairs per rule.

E.2 Electricity Protocol and Diagnostics

In this case study, we use historical Christmas patterns to select a single-feature edit for a forecast issued on Christmas Eve. We compare the resulting forecasts with native Chronos-2 and a calendar-based historical-ratio baseline.

Data and Discovery. We use the hourly GIFT-Eval electricity/H data derived from UCI ElectricityLoadDiagrams20112014, without further resampling. For 2011–12, discovery compares each meter's 512-hour histories ending December 25 and December 18 at 23:00. The mean final-token code difference $\overline{\Delta h}$ ranks features by absolute magnitude; the top 16 enter selection. Fig. 14 displays the top eight.

For candidate j, let $\boldsymbol{r}_j = D(\overline{\Delta h}_j \boldsymbol{e}_j) - D(\mathbf{0})$. Each candidate is rescaled to the norm of the first-ranked direction, feature 525:

$$\boldsymbol{d}_j = \frac{\|\boldsymbol{r}_{525}\|_2}{\|\boldsymbol{r}_j\|_2} \boldsymbol{r}_j. \tag{9}$$

This gives equal per-token edit norms across candidates at a common gain. The selected edit adds $8\boldsymbol{d}_{2735}$ at context positions 24–31, regardless of the feature's original activation. Register and future-patch states are unchanged at the intervention layer.

Selection and Evaluation. Selection compares 16 features, gains $\alpha \in \{\pm 2, \pm 4, \pm 8, \pm 16, \pm 32\}$, and widths 1 or 8 (320 configurations). The 343 valid 2013 meters are split by the parity of the

first SHA-256 byte of their IDs: 166 for selection and 177 for a separate evaluation. Feature 2735, gain 8, and width 8 minimize selection nMAE. Native/edit/ratio scores are 0.260/0.158/0.216 on the selection split and 0.253/0.145/0.194 on the remaining meters.

Eligible windows have finite observations and context/reference means above 5. Two meters with nonfinite native forecasts are excluded in each year, leaving 366 in 2014, including all 343 from 2013. The input ends December 24 at 23:00; the target is December 25, 00:00–23:00. For context mean a_m and 24-hour target $y_{m,t}$, we report

$$a_m = \frac{1}{512}\sum_{u=1}^{512} s_{m,u}, \qquad \mathrm{nMAE} = \frac{1}{366}\sum_{m=1}^{366}\frac{1}{24a_m}\sum_{t=1}^{24}|\widehat{y}_{m,t} - y_{m,t}|. \tag{10}$$

Errors are computed per meter, not from the plotted mean profile. The edit improves 303 meters over native and 264 over historical ratio. A 2,000-replicate meter bootstrap (seed 251225) gives an edit-minus-native interval of $[-0.1482, -0.1157]$.

Table 8: **Christmas forecast performance.** Results across 366 meters.

Metric	Native	SAE edit	Historical ratio
Mean nMAE	0.3233	0.1913	0.2435
Median per-meter nMAE	0.2463	0.1262	0.1760

Historical-ratio Baseline. The plotted historical-ratio baseline multiplies the observed December 18, 2014 hourly profile by

$$R_m = \mathrm{median}_{y\in\{2011,2012\}}\,\mathrm{clip}_{[0.2,2]}\left(\frac{\overline{y}_{m,y,\mathrm{Dec\,25}}}{\overline{y}_{m,y,\mathrm{Dec\,18}}}\right), \tag{11}$$

where bars denote daily means and only valid historical ratios are used. Of 366 meters, 328 have their own ratio; the remaining 38 use the pooled median 0.6134.

Feature and Gain Diagnostics. Feature 2735 survives 275 of 298 Hop-2 perturbations and 278 of 298 Hop-3 perturbations (critical count 277; null survival probability 0.9; significance 0.05). It activates in 34 of 366 unedited contexts and in the final eight positions for 11. A gain-sensitivity sweep shows stable performance around the selected gain $\alpha = 8$.

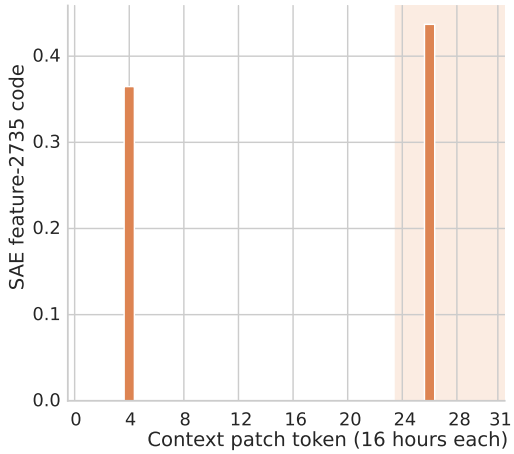


Figure 13: **Feature 2735 across the observed context.** SAE codes over the 32 context-patch states of meter MT_134. Shading marks the final eight states used in the steering example.

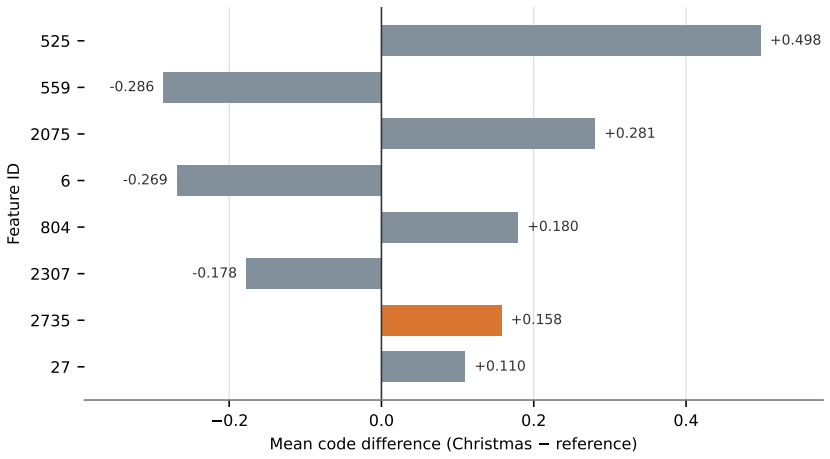


Figure 14: **Historical feature discovery.** Top eight SAE features with the strongest Christmas/reference activation shifts in 2011–12. Feature 2735, later selected for steering, is highlighted.

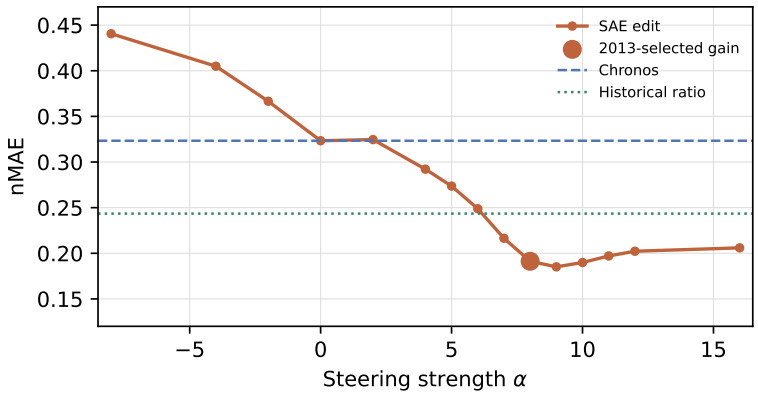


Figure 15: **Gain sensitivity on 2014.** The marker denotes the 2013-selected gain $\alpha = 8$.

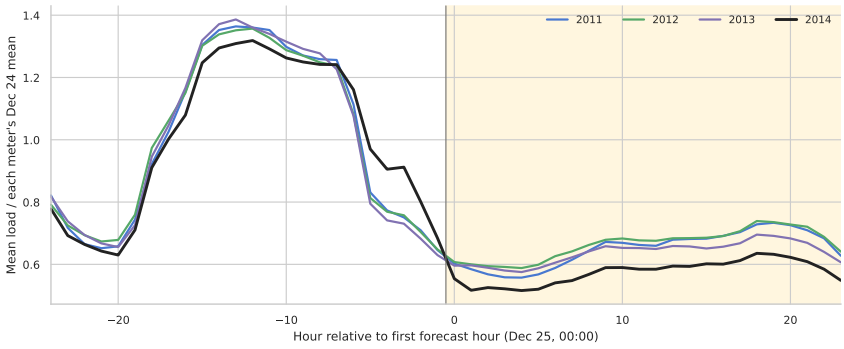


Figure 16: **Observed Christmas profiles, 2011–14.** Mean load over 158 common meters, normalized by each meter-year's December 24 mean. Shading marks December 25.