TSAE: STRUCTURED SPARSE AUTOENCODERS FOR INTERPRETING TIME-SERIES FORECASTING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series forecasting informs critical decisions in energy dispatch, industrial operations, and environmental monitoring; understanding the patterns models rely on is essential for assessing reliability and identifying failures. Input attribution identifies important variables and time segments but offers limited insight into internal features, while standard sparse autoencoder (SAE) objectives do not directly constrain cross-variable structure or temporal continuity. We introduce TSAE, a structured sparse autoencoder for forecasting representations that decomposes hidden states into individually inspectable features. TSAE organizes cross-variable structure through shared and variable-routed private dictionaries, separates feature detection from magnitude estimation with gated encoding, and constrains neighboring sparse-code changes according to raw-segment similarity. These mechanisms support analysis of variable context, activation strength, and temporal evolution. Forecast-consistency fine-tuning further improves preservation of the frozen forecaster's outputs. The accompanying TSEVAL protocol separately audits fidelity, feature coherence, and physical calibration to ground feature interpretation. In three-seed experiments with frozen PatchTST on ETTh1, ETTh2, and ETTm1, TSAE achieves the lowest hidden-state reconstruction error, normalized forecast-reconstruction error (NFRE), and feature transition rate among five SAEs at comparable per-token activity. NFRE decreases by 4.3–26.4% relative to the next-best mean. Dataset-dependent tradeoffs in selectivity, physical correlation, and calibration show that fidelity and temporal-stability gains require independent semantic validation to support feature interpretation.

1 INTRODUCTION

Time-series interpretability is essential for deploying forecasting models in high-stakes domains such as energy, climate, healthcare, and finance. In these settings, prediction errors can lead to substantial real-world costs, making model reliability crucial.

Most widely used time-series interpretability methods explain forecasts at the input level. Saliency maps, occlusion analysis, Shapley values, dynamic masking, and counterfactual explanations attribute predictions to variables, timesteps, or perturbed subsequences (Sundararajan et al., 2017; Bento et al., 2021; Crabbé & van der Schaar, 2021). However, input-level attribution alone does not probe the latent features learned by the forecasting backbone, making it difficult to assess whether the model encodes semantically coherent, physically grounded, and forecast-relevant temporal patterns.

Recently, Sparse Autoencoders (SAEs) have become an increasingly used approach for interpreting hidden representations in large-scale models. By decomposing hidden states into sparse linear combinations of learned feature directions (Bricken et al., 2023; Gao et al., 2025), SAEs can help identify recurring internal concepts beyond input-level attribution. While established practices in language model interpretability—such as assessing reconstruction fidelity, measuring sparsity, and analyzing top-activating examples—are well-defined, their application to time-series remains underexplored. This gap highlights a significant opportunity to leverage SAEs for time-series interpretability, potentially exposing reusable, semantically meaningful hidden features within forecasting backbones.

However, standard SAEs are not designed to explicitly model the time-series structure illustrated in Figure 1. In multivariate forecasting, dependencies across variates can make shared temporal

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.

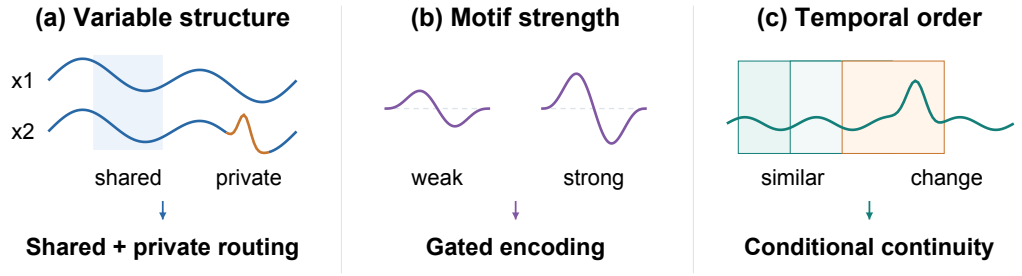


Figure 1: Three time-series properties motivate TSAE: (a) shared motifs and variable-specific structure motivate shared/private routing; (b) motif strength motivates separating detection from magnitude; (c) similar adjacent patches of one variable motivate code continuity, with weaker coupling across real changes. Traces are schematic.

patterns relevant (Liu et al., 2024), while patch-based models construct neighboring tokens from overlapping local segments (Nie et al., 2023). When a backbone exposes variable-indexed, ordered tokens, a useful time-series SAE should distinguish meaningful cross-variable sharing from uncontrolled mixing and encourage stable feature identities across adjacent patches when their raw segments are similar. Separately, because sparsity penalties can shrink activation magnitudes (Rajamanoharan et al., 2025), it is important to audit rather than assume whether sparse activations remain calibrated to local signal variation. A flat reconstruction–sparsity objective does not directly encourage these properties.

Beyond learning sparse features, time-series interpretability also requires evaluating whether those features are meaningful for forecasting. Reconstruction loss and top-activating examples indicate that a dictionary can preserve hidden states and expose inspectable patterns, but they do not show whether a feature is coherent across variables, calibrated to signal motifs, or relevant to forecasting. We therefore need an evaluation protocol that separates reconstruction fidelity, feature purity, physical calibration, forecast sensitivity, and input-level baselines.

To address these limitations, we propose TSAE (Time-Series Structured Sparse Autoencoder), a structured sparse autoencoder for interpreting time-series forecasting representations. TSAE combines shared and variable-routed private dictionaries with mechanisms for magnitude preservation and temporal stability, encouraging a separation between shared temporal motifs and variable-conditioned residual structure. Complementing the model, we introduce TSEVAL, a capability-graded multi-axis diagnostic protocol that evaluates sparse features beyond reconstruction, including purity, physical calibration, and forecast intervention sensitivity.

Our contributions are:

- We introduce TSAE, a structured SAE for variable-indexed and time-ordered forecasting representations, together with an inductive-bias analysis explaining how shared/private dictionaries, magnitude preservation, and temporal regularization are designed to address limitations of flat reconstruction–sparsity objectives.
- We develop TSEVAL, a six-axis evaluation protocol and adapter toolkit for auditing sparse features in terms of reconstruction, purity, physical calibration, forecast sensitivity, hidden-space intervention, and input-level baselines.
- We validate TSAE with active-matched comparisons across multiple datasets, forecasting backbones, and independent random seeds, together with component ablations that examine routing, gating, and temporal continuity.

Related work is discussed in Appendix B.

2 TSAE: Structured Sparse Autoencoder

TSAE is a structured sparse autoencoder inserted at the representation level of a forecasting model. A raw multivariate sequence is first encoded by a forecasting backbone into hidden tokens $h_{b,c,p} \in$

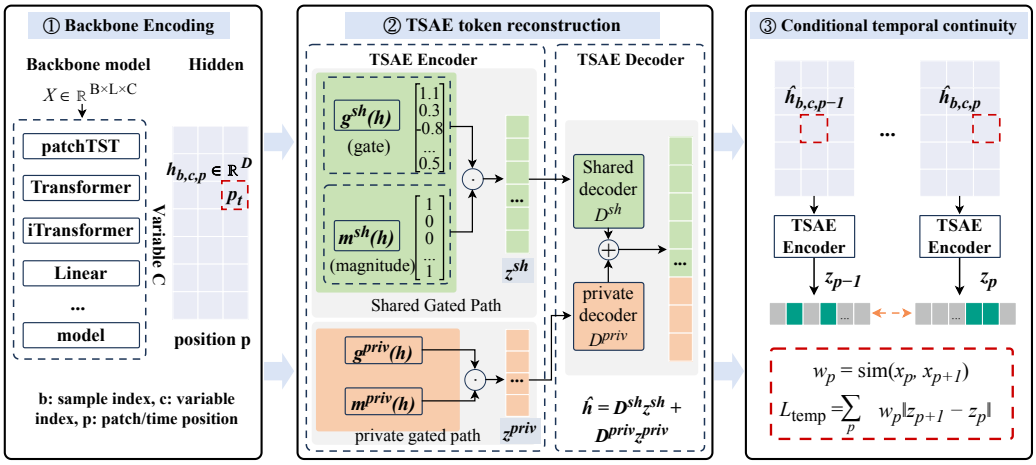


Figure 2: TSAE model framework. Raw time-series inputs are first encoded by a forecasting backbone into hidden tokens. For each token, the TSAE encoder produces shared and variable-routed private gated sparse codes, and the decoder reconstructs the hidden token from the corresponding shared and private dictionaries. Conditional temporal continuity is applied to adjacent sparse codes, with the regularization strength determined by the similarity of their raw temporal segments.

$\mathbb{R}^D$, and TSAE then learns an overcomplete sparse code and reconstructs the hidden token, not the raw input series. We do not assume that hidden states admit a unique decomposition into ground-truth factors; instead, we use three task-specific inductive biases: variable structure, magnitude preservation, and temporal structure. These biases motivate shared/private variable routing, gated detection–magnitude separation, and conditional temporal continuity, respectively.

Shared/private variable routing. Let $h_{b,c,p} \in \mathbb{R}^D$ denote a hidden token from sample b, variable c, and token position p. A flat SAE maps $h_{b,c,p}$ to an overcomplete sparse vector $z \in \mathbb{R}^K$ and decodes it back to $\hat{h}_{b,c,p} \in \mathbb{R}^D$ with one shared dictionary. Cross-variable activation is not inherently problematic when a feature consistently captures the same temporal motif across variables; the difficulty is that the same dictionary can also reuse one sparse direction for different motifs under different variable contexts.

TSAE therefore reconstructs each token with a shared component and a variable-routed private component:

$$\hat{h} = D^{\mathrm{sh}} z^{\mathrm{sh}} + D^{\mathrm{priv}}_c z^{\mathrm{priv}} + b, \tag{1}$$

where $z^{\mathrm{sh}} \in \mathbb{R}^{K_{\mathrm{sh}}}$, $z^{\mathrm{priv}} \in \mathbb{R}^{K_{\mathrm{priv}}}$, D^{sh} is shared across variables, and D^{priv}_c is selected by the variable index c. Thus the SAE reconstruction step becomes $\mathbb{R}^D \to \mathbb{R}^{K_{\mathrm{sh}}+K_{\mathrm{priv}}} \to \mathbb{R}^D$ for each routed token, with private capacity kept compact. The split is learned rather than assigned by a hard rule: each token can activate both paths, while reconstruction, sparsity, and limited private capacity encourage recurring cross-variable motifs to use shared features and variable-conditioned residual structure to use private features. This induces, rather than guarantees, a separation between meaningful sharing and uncontrolled mixing.

Gated detection and magnitude. The gated step occurs inside the encoder when $h \in \mathbb{R}^D$ is lifted into the overcomplete sparse space. Each shared or private feature first computes a learned direction and then applies separate gate and magnitude transformations to that direction:

$$z = \mathbf{1}[g(h) > 0] \odot m(h), \tag{2}$$

Here $g(h)$ detects whether the feature is present, $m(h)$ is a nonnegative magnitude estimate, and $\odot$ denotes elementwise multiplication. If the gate is closed, the feature activation is zero; if it is open, the activation takes the magnitude value. TSAE applies this parameterization to both the shared path, producing z^{sh}, and the variable-routed private path, producing z^{priv}. The sparsity penalty is applied to gate activations rather than directly to magnitudes. This follows the Gated SAE idea of separating feature selection from feature magnitude, but adapts it to time-series interpretation: the

model is encouraged to open fewer features, while the magnitude path can still preserve feature strength for comparison with physical motifs such as slope or volatility. Excessive sparsity-driven shrinkage of magnitudes would weaken the link between sparse activations and signal strength.

Conditional temporal continuity. The temporal term is applied after the encoder has produced sparse codes, not as a separate reconstruction path. For two neighboring ordered temporal tokens from the same sample and variable, TSAE first computes $z_{b,c,p} = [z^{\mathrm{sh}}_{b,c,p}, z^{\mathrm{priv}}_{b,c,p}]$ and $z_{b,c,p+1}$ through the same shared/private gated encoder. If the corresponding raw tokens are similar, abrupt changes in sparse feature identity can reflect feature flicker rather than meaningful temporal change; if the raw signal contains spikes or regime changes, sparse features should be allowed to change. TSAE therefore applies temporal smoothing conditionally: feature differences are penalized strongly only when the corresponding raw tokens are similar.

For adapters that expose multiple ordered temporal tokens per variable, we build a chain graph from adapter metadata rather than inferring adjacency from hidden-space similarity. Tokens are grouped by sample and variable, sorted by token position, and adjacent positions are connected. We then apply a weighted graph total variation:

$$L_{\mathrm{temp}} = \mathbb{E}_{b,c,p}\left[w_{b,c,p}\left\|z_{b,c,p+1} - z_{b,c,p}\right\|_1\right]. \tag{3}$$

The weight $w_{b,c,p} \in [0,1]$ is an RBF similarity between adjacent locally normalized raw tokens:

$$w_{b,c,p} = \exp\left(-\frac{\mathrm{dist}(\bar{x}_{b,c,p+1}, \bar{x}_{b,c,p})^2}{2\sigma^2}\right), \tag{4}$$

Here $\bar{x}$ denotes raw patches normalized jointly within each sample–variable chain, $\mathrm{dist}(u,v) = \|u - v\|_2/\sqrt{\ell}$ is the normalized Euclidean distance between two segments of length ℓ, and the PatchTST headline runs use $\sigma = 1$. Thus, similar raw tokens yield large weights and strongly discourage unnecessary feature switching, whereas dissimilar tokens yield small weights and allow activations to change for spikes or regime transitions. For adapters without ordered temporal edges, such as single-token or non-temporal-token representations, the temporal graph is empty and $L_{\mathrm{temp}} = 0$.

Overall objective. The first training stage jointly optimizes the three mechanisms with a compact objective:

$$\begin{aligned} L_{\mathrm{base}} = L_{\mathrm{recon}} + L_{\mathrm{sp}} + \lambda_{\mathrm{gate}} L_{\mathrm{gate}} \\ + \lambda_{\mathrm{temp}} L_{\mathrm{temp}} + L_{\mathrm{aux}}. \end{aligned} \tag{5}$$

Here L_{recon} reconstructs forecasting hidden tokens and $L_{\mathrm{sp}} = \lambda_{\mathrm{shared}}\,\overline{a}^{\mathrm{sh}} + \lambda_{\mathrm{priv}}\,\overline{a}^{\mathrm{priv}}$, where $\overline{a}^{\mathrm{sh}}$ and $\overline{a}^{\mathrm{priv}}$ are the mean nonnegative gate signals in the shared and private paths. L_{gate} is the gate-reconstruction auxiliary loss, and L_{temp} is the conditional temporal continuity term. The auxiliary term is $L_{\mathrm{aux}} = \lambda_{\mathrm{balance}} L_{\mathrm{balance}} + \lambda_{\mathrm{orth}} L_{\mathrm{orth}}$: L_{balance} discourages private-feature usage from collapsing within a variable, while L_{orth} discourages redundant shared and private decoder directions.

Forecast-consistency fine-tuning. The complete TSAE training protocol then performs a short second-stage refinement through the frozen forecasting backbone. Let f denote the fixed forecasting map from the adapted hidden representation to the target forecast, and let $\hat{h}$ be the TSAE reconstruction of h. We optimize only TSAE parameters with

$$L_{\mathrm{FT}} = L_{\mathrm{base}} + \lambda_{\mathrm{fc}} \frac{\mathrm{MSE}(f(\hat{h}), f(h))}{\mathrm{Var}(f(h))}. \tag{6}$$

This forecast-consistency term encourages the reconstructed hidden representation to preserve the frozen backbone's output while retaining the first-stage reconstruction, sparsity, auxiliary, and temporal objectives. In the PatchTST headline runs, $\lambda_{\mathrm{fc}} = 0.1$, and this stage uses two epochs with at most 80 training batches per epoch. It is a component of the complete TSAE method; the forecasting backbone itself is never updated.

This objective is intended as an inductive bias rather than a guarantee of unique factor recovery. It encourages reusable temporal motifs to be represented by shared gated features, variable-conditioned residual structure by compact private features, and similar adjacent ordered temporal tokens by stable sparse codes. We therefore evaluate TSAE as a structured alternative to flat shared SAEs, rather than claiming uniform dominance on every metric.

3 TSEVAL: TIME-SERIES SAE EVALUATION PROTOCOL

TSEVAL is a companion protocol for auditing the sparse features learned by TSAE and baseline SAE variants. Its role is deliberately narrower than the method section: TSAE defines how sparse features are learned, while TSEVAL defines which interpretability claims those features support. The protocol evaluates six axes, each targeting a failure mode that reconstruction-only SAE evaluation cannot rule out: reconstruction fidelity, feature coherence, physical calibration, forecast intervention sensitivity, hidden-space controls, and input-level baselines. Table 1 summarizes the diagnostic question associated with each axis.

Diagnostic	**Question**
Fidelity	Does the sparse reconstruction preserve the original hidden state?
Coherence	Are activations concentrated in coherent variable and temporal contexts?
Calibration	Do feature activations track independently measurable signal motifs?
Forecast sensitivity	How sensitive is the forecast to a specified sparse-feature intervention?
Hidden control	Is the effect specific to the SAE decoder path rather than a generic hidden-space perturbation?
Input baselines	Do hidden sparse features provide information beyond saliency and occlusion?

Table 1: The TSEVAL diagnostic checklist. Each axis tests a claim that reconstruction-only evaluation cannot adjudicate. In TSEVAL, a feature is interpreted only with respect to the claims supported by the corresponding diagnostics.

A complete TSEVAL run takes a frozen forecasting backbone, hidden tokens with metadata, an SAE configuration, and raw segments for physical-motif computation. It then adapts hidden states to a common token contract; trains or loads the sparse dictionary; scores fidelity, coherence, and physical calibration; measures forecast sensitivity through decoder-mediated feature ablations and hidden-direct controls; and compares hidden-feature signals with saliency and occlusion baselines. Coherence is operationalized by semantic/channel concentration statistics, including purity scores for compatibility with prior SAE analyses.

Diagnostic definitions. Fidelity is the SAE reconstruction error on frozen backbone hidden tokens. Calibration selects the strongest of eight raw-segment motifs and reports standardized-fit residuals as calibration error. Forecast sensitivity is measured by decoder-mediated ablation: with $z = E(h(x))$, $\hat{h} = D(z)$, and $\hat{h}^{(-i)} = D(z_{\setminus i})$,

$$\mathrm{FIS}_i = \mathbb{E}_{(x,y)} \left[\ell(f(\hat{h}^{(-i)}), y) - \ell(f(\hat{h}), y) \right]. \tag{7}$$

The hidden-direct control subtracts the matched direction $z_i d_i$ from the original hidden state to test whether the forecast change is specific to the SAE decoder path. FIS is therefore an intervention-sensitivity diagnostic, not a causal effect estimator.

Capability-graded execution. Adapters normalize backbone-specific representations into $H \in \mathbb{R}^{B\times C\times D\times P}$ when the corresponding axes are exposed, and record capability flags for decoding, temporal order, hidden controls, decoder marks, and cycle indices. Diagnostics are enabled only when the required token axes and decoding capabilities are available; otherwise the axis is marked unsupported rather than inferred from artificial metadata. The released toolkit covers 18 registered forecasting models through five adapter families, but smoke tests validate plug-in coverage and reproducibility, not scientific performance.

Why multiple axes are necessary. The axes are intentionally non-equivalent. Fidelity does not distinguish a meaningful shared temporal motif from uncontrolled cross-variable mixing. Coherence can reflect contextual concentration without tracking any physical motif. Calibration can align a feature with slope or volatility even when the forecasting head is insensitive to that decoder direction.

TSEVAL therefore treats reconstruction as a precondition, physical correlation as semantic alignment rather than forecast causality, and FIS as forecast sensitivity under a specified intervention. Any feature-level interpretation should be restricted to the claims supported by its corresponding diagnostics.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Backbone, datasets, and baselines. We evaluate TSAE on frozen PatchTST patch-token representations. The core benchmark compares TSAE with L1, Gated, TopK, and JumpReLU SAEs on ETTh1, ETTh2, and ETTm1 under common data splits and a 2,048-feature per-token encoding width. This matches per-token code width, not the total routed dictionary parameter count. TSAE's complete protocol includes the forecast-consistency refinement in Section 2; baselines retain their original SAE objectives, while all forecasting backbones remain frozen. Weather is an external-validity extension. Validation data select comparable operating points, and the PatchTST–ETTh1 validation sweep compares fidelity against realized activity. Main-table results are three-seed means $\pm$ standard deviations; iTransformer and DLinear are transfer checks.

Evaluation protocol. For the PatchTST–ETTh1 activity–fidelity analysis, we sweep each method's sparsity controls on validation data over a broad range of realized activity levels (roughly 10^2–10^3 active features per token where supported by the method). Fidelity–sparsity tradeoffs are compared using measured activity, rather than raw hyperparameter values. For the formal three-seed PatchTST comparison, we fix a validation-selected configuration within a prespecified comparable activity band before retraining from scratch on the training split and evaluating each seed once on the test split. TopK uses an exact integer activity budget. The full selection rule and selected operating points are reported in the appendix.

The PatchTST core evaluation reports reconstruction MSE, normalized forecast-reconstruction error (NFRE), mean active features, dead-feature ratio, ordered-code transition rate, selectivity AUROC, physical motif correlation, and calibration error. Reconstruction MSE and NFRE quantify forecast-relevant fidelity; mean activity and dead-feature ratio measure sparsity and dictionary utilization; the remaining diagnostics assess temporal stability, selectivity, physical alignment, and calibration. Cross-backbone transfer checks report the subset of metrics that is comparably defined for their representation geometry.

4.2 MAIN RESULTS

Our empirical evidence is organized around a controlled PatchTST comparison and two complementary analyses. The headline evaluation fixes the forecasting representation and compares five SAE methods on ETTh1, ETTh2, and ETTm1 under the validation-selected, three-seed test protocol; Weather is reported under the same protocol as an external-validity extension. We then test whether the fidelity pattern persists across realized activity levels with a validation-only curve, and isolate the contributions of routing, gating, and temporal continuity by ablation. The appendix records validation selection, full configurations, and the iTransformer and DLinear extensions; these extensions are supporting evidence rather than replacements for the controlled comparison.

Controlled PatchTST comparison. Table 2 reports the complete five-method comparison across four datasets. TSAE has the lowest reconstruction MSE and NFRE on all four. On the three ETT datasets, its NFRE reductions relative to the next-best mean are 4.3% (ETTh1), 26.4% (ETTh2), and 9.1% (ETTm1); on Weather the reduction is 11.3%. Reconstruction MSE follows the same ordering, with reductions of 1.0–24.4%. Operating points are selected under the prespecified validation active-feature window. For the three ETT datasets, every reported test mean lies in $[192, 320]$. Weather TSAE, Gated, TopK, and JumpReLU also remain in-window; the Weather L1 test mean is slightly below the boundary (187.8) despite its validation-selected point being in-window. The active-feature protocol controls realized per-token activity; total routed dictionary parameter counts are reported separately in the appendix.

Setting		Operating point	Evaluation metrics						
Data	Method	Active	Recon. ↓	NFRE ↓	Dead ↓	Trans. ↓	Sel. ↑	Phys. ↑	Calib. ↓
ETTh1	TSAE	231.4±2.7	**13.44±0.57**	**5.10±0.25**	0.080±0.025	**0.073±0.005**	**0.639±0.007**	0.541±0.106	0.624±0.049
	L1	252.8±10.3	13.59±0.45	5.33±0.27	**0.000±0.000**	0.181±0.006	0.570±0.023	**0.653±0.047**	**0.567±0.022**
	Gated	244.4±7.4	16.39±0.75	6.39±0.22	0.000±0.000	0.144±0.005	0.573±0.020	0.523±0.060	0.681±0.024
	TopK	264.9±5.7	30.54±2.00	10.84±0.75	0.001±0.001	0.143±0.002	0.592±0.007	0.622±0.047	0.587±0.043
	JumpReLU	269.8±11.4	20.69±0.88	8.09±0.37	**0.000±0.000**	0.156±0.007	0.612±0.066	0.587±0.017	0.657±0.002
ETTh2	TSAE	250.9±13.6	**6.04±0.29**	**2.70±0.14**	0.043±0.014	**0.050±0.002**	0.327±0.001	0.571±0.066	0.595±0.027
	L1	195.2±11.4	7.99±0.18	3.66±0.23	**0.025±0.016**	0.106±0.009	**0.335±0.005**	0.600±0.087	0.581±0.036
	Gated	212.7±6.0	8.19±0.46	3.76±0.20	0.103±0.026	0.055±0.003	0.327±0.005	0.604±0.040	0.593±0.016
	TopK	218.3±1.0	9.88±0.37	4.56±0.20	0.143±0.016	0.063±0.003	0.329±0.004	0.498±0.047	0.671±0.027
	JumpReLU	311.0±39.9	10.86±0.74	4.79±0.27	0.041±0.015	0.064±0.007	0.333±0.006	**0.752±0.048**	**0.520±0.025**
ETTm1	TSAE	204.9±11.5	**10.35±0.48**	**5.00±0.28**	0.031±0.004	**0.047±0.002**	0.615±0.032	0.571±0.068	0.673±0.036
	L1	258.0±11.9	11.44±0.65	5.50±0.25	**0.000±0.000**	0.135±0.005	0.620±0.038	0.565±0.127	**0.608±0.055**
	Gated	246.9±5.8	13.49±0.80	6.58±0.41	0.000±0.000	0.089±0.002	0.617±0.007	0.548±0.033	0.677±0.022
	TopK	223.7±2.4	19.92±0.59	8.02±0.16	0.006±0.001	0.083±0.001	**0.638±0.013**	**0.598±0.050**	0.623±0.025
	JumpReLU	269.9±12.8	17.09±0.97	8.63±0.27	**0.000±0.000**	0.092±0.004	0.630±0.019	0.568±0.108	0.669±0.024
Weather	TSAE	216.7±3.2	**1.70±0.06**	**2.52±0.22**	0.195±0.006	**0.019±0.000**	0.574±0.016	0.310±0.016	0.768±0.006
	L1	187.8±16.3	1.85±0.15	3.37±0.22	**0.000±0.000**	0.055±0.003	0.557±0.005	0.310±0.005	**0.725±0.006**
	Gated	285.1±8.8	2.10±0.15	2.84±0.13	0.000±0.000	0.046±0.002	0.571±0.021	0.319±0.014	0.765±0.007
	TopK	261.1±14.4	3.84±0.15	4.64±0.35	0.073±0.004	0.038±0.002	0.533±0.018	0.312±0.010	0.755±0.004
	JumpReLU	236.0±0.7	4.49±0.53	6.21±0.84	0.001±0.001	0.031±0.002	**0.577±0.016**	**0.321±0.006**	0.765±0.008

Table 2: Validation-selected PatchTST results on four datasets. Each entry is the mean±standard deviation over three independently trained SAE seeds. Operating points are selected only on validation data to target 256 active features per token (admissible range: 192–320); all reported metrics are test-split measurements. Bold and underline mark the best and second-best mean, respectively, within a dataset and metric. Recon. and NFRE are multiplied by 10^3.

Activity–fidelity analysis. Table 2 compares methods at validation-selected operating points, but does not show whether the fidelity advantage is confined to one realized activity level. We therefore sweep SAE sparsity controls on the PatchTST–ETTh1 validation split and plot fidelity against measured active features per token in Figure 3. This validation analysis complements, rather than replaces, the test-split comparison in Table 2.

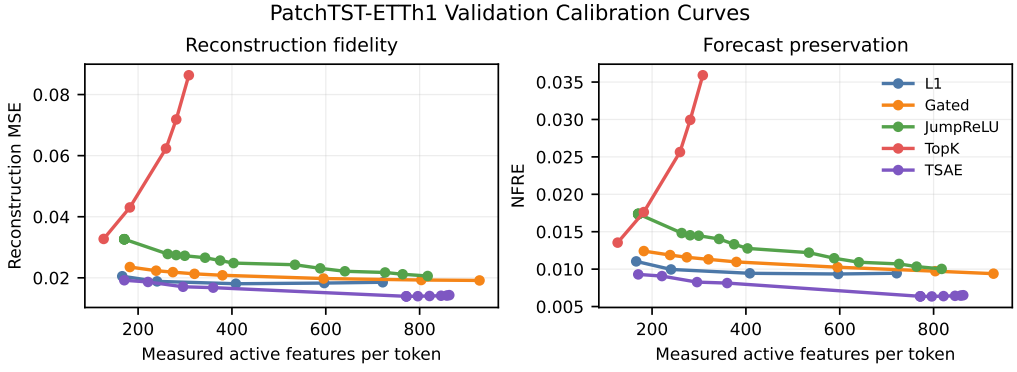


Figure 3: PatchTST–ETTh1 validation activity–fidelity curves. Each point is a validation calibration run; the horizontal axis reports measured active features per token. Table 2 reports separate test-split metrics.

In the sampled overlapping activity range, TSAE has lower reconstruction and forecast-reconstruction error than the displayed baselines. Its Table 2 fidelity advantage is therefore not tied to a single selected activity level. Because Figure 3 is a single-dataset validation analysis, we treat it as supporting robustness evidence rather than an additional headline result.

To test whether the fidelity advantage transfers beyond PatchTST, we ran the same active-matched protocol on frozen iTransformer–ETTh1 and DLinear–ETTh1 representations. Table 3 summarizes the results with the PatchTST comparison for reference.

Backbone	Method	Active	Recon. ↓	NFRE ↓
PatchTST	TSAE	231.4±2.7	**13.44±0.57**	**5.10±0.25**
	Gated	244.4±7.4	16.39±0.75	6.39±0.22
	L1	252.8±10.3	13.59±0.45	5.33±0.27
	TopK	264.9±5.7	30.54±2.00	10.84±0.75
	JumpReLU	269.8±11.4	20.69±0.88	8.09±0.37
iTransformer	TSAE	253.1±2.5	**14.48±0.23**	**3.01±0.06**
	Gated	315.7±1.1	17.29±0.13	3.20±0.09
	L1	282.3±1.0	18.12±0.08	4.14±0.20
	TopK	211.8±0.3	30.10±0.14	5.24±0.15
	JumpReLU	268.6±3.4	18.84±0.11	3.44±0.05
DLinear	TSAE	215.0±10.2	9.09±0.16	7.91±0.23
	Gated	314.1±1.1	**7.71±0.09**	**6.88±0.14**
	L1	250.4±1.2	9.17±0.09	8.74±0.14
	TopK	256.0±0.0	9.90±0.21	8.27±0.22
	JumpReLU	266.0±3.7	7.99±0.12	7.02±0.16

Table 3: Cross-backbone active-matched comparison (ETTh1, three-seed means). Bold and underline mark the best and second-best means, respectively, within each backbone and metric. Recon. and NFRE are $\times 10^3$. Full tables and validation sweeps are in Appendix C.4.

Cross-backbone diagnostic. On PatchTST and iTransformer, TSAE yields the lowest reconstruction and NFRE, consistent with the primary result. On DLinear, Gated SAE has the best reconstruction and NFRE, indicating that the fidelity advantage does not automatically transfer to every architecture. These cross-backbone comparisons should be interpreted as supporting evidence rather than as replacements for the controlled PatchTST headline evaluation.

This fidelity advantage does not imply uniform dominance on every diagnostic. L1 has the lowest dead-feature ratio and calibration error on several datasets; TopK and JumpReLU lead selectivity or physical correlation in some settings. TSAE instead makes a narrower claim: under a controlled activity budget, its structured code preserves forecast-relevant hidden information best and yields the lowest transition rate in the four reported PatchTST comparisons. The appendix contains the selected settings and validation-sweep record.

4.3 Ablation Study

The ablation examines the behavior of TSAE components on the PatchTST backbone with ETTh1, ETTh2, and ETTm1. We compare the full model with variants that remove private routing, gated parameterization, or the temporal-continuity term. The three-seed results in Table 4 summarize changes in activity, fidelity, dictionary utilization, and temporal stability. They are component diagnostics, rather than capacity- or activity-matched causal estimates.

Component contributions. In the reported configurations, removing routing raises transition rate by 39–69% across the three datasets (ETTh2: 0.036→0.050; ETTm1: 0.049→0.071; ETTh1: 0.086→0.145). Removing gating reduces realized activity from 203–268 to 46–108 and coincides with worse reconstruction, so it does not isolate a fidelity effect at matched activity. The temporal term has a smaller anti-flicker effect: its validation-selected coefficient is $\lambda_{\text{temp}} = 0.5$, whereas $\lambda_{\text{temp}} = 0.001$ is indistinguishable from zero in the sensitivity diagnostic.

Mechanism. These directions are consistent with the structural roles in Section 2: routing allocates shared motifs and variable-conditioned features separately, gating separates detection from magnitude, and temporal continuity discourages switching on similar adjacent patches. The ablation alone does not identify these mechanisms apart from changes in total capacity or realized activity.

Data	Variant	Active	Recon. ↓	NFRE ↓	Dead	Trans.
ETTh1	Full	268	**11.7**	**4.9**	6.3	0.086
	No routing	252	13.4	5.5	0.0	0.145
	No gating	108	18.1	7.9	0.0	0.083
	No temporal	279	12.3	5.1	6.3	0.091
ETTh2	Full	203	**2.2**	**3.4**	10.9	0.036
	No routing	178	2.4	3.6	13.1	0.050
	No gating	46	3.7	5.7	23.3	0.027
	No temporal	234	2.8	4.2	13.3	0.038
ETTm1	Full	218	**9.1**	**5.2**	3.8	0.049
	No routing	192	9.8	5.7	0.1	0.071
	No gating	79	14.1	8.5	0.0	0.047
	No temporal	261	10.3	5.8	6.2	0.052

Table 4: PatchTST component ablation (three-seed means). Bold marks the lowest reconstruction and NFRE values within each dataset. Dead-feature ratio and transition rate are reported as diagnostic tradeoffs. Recon. and NFRE use the report scale ($\times 10^3$).

5 Limitations

The main evidence is strongest for PatchTST on the three ETT datasets. This is a deliberate choice: it lets us compare sparse-feature methods under a controlled hidden representation and a dataset-specific active-feature budget. It also leaves open how far the same design transfers to other forecasting architectures and higher-dimensional datasets; those questions require a separate, fully matched evaluation.

The diagnostic metrics should also be read separately. TSAE is most consistent on reconstruction and NFRE, but it is not uniformly best on every interpretability proxy. JumpReLU or TopK can be stronger on selectivity, physical correlation, or calibration error, depending on the dataset. Physical motifs are hand-designed shape statistics, so they may miss domain-specific semantics. FIS-style interventions measure sensitivity under a specified SAE decoder intervention; they are not exhaustive causal attribution. Finally, conditional temporal continuity is meaningful only when the adapter exposes ordered temporal tokens. For non-patch or unordered token representations, the temporal graph is empty and the term should not be overinterpreted.

6 Conclusion

We introduced TSAE, a structured sparse autoencoder for forecasting representations. Its shared/private dictionaries separate reusable temporal motifs from variable-conditioned residual structure, its gated parameterization protects activation magnitudes from sparsity-induced shrinkage, and its conditional temporal term discourages feature flicker on similar adjacent patches. Under an active-matched protocol on ETTh1, ETTh2, and ETTm1, TSAE achieves the lowest reconstruction error and NFRE among the five compared methods on all three datasets.

The result is not a blanket dominance claim. The stronger conclusion is that complete TSAE improves fidelity to forecast-relevant hidden information while exposing real tradeoffs with selectivity, physical correlation, and calibration. The reported ablations are consistent with routing improving temporal stability, gating avoiding low realized activity, and temporal continuity reducing patch-token flicker. These findings support TSAE as a practical, structure-aware alternative to flat SAEs for time-series forecasting interpretation.

AI USE STATEMENT

The authors used large language models (LLMs) as a research aid in preparing this paper. LLM assistance covered writing and polishing portions of the text, writing and debugging analysis code, brainstorming experimental designs and interpreting experimental results, and discovering relevant literature. All LLM-assisted output was verified and curated by the authors, who take full responsibility for the content of this paper. LLMs were not used to fabricate results and are not listed as authors.

REPRODUCIBILITY STATEMENT

All headline comparisons use three independently initialized SAE runs; run-level values, operating points, and audit trails are reported in the appendices. Code, configurations, and the data-processing pipeline will be released publicly with a research-use license upon publication. Appendix C documents the evaluation protocol and Appendix C.5 the computational environment.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018.

João Bento, Pedro Saleiro, André F. Cruz, Mário A. T. Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023.

Bart Bussmann, Michael Pearce, Lee Sharkey, Dan Braun, and Neel Nanda. Matryoshka sparse autoencoders. *arXiv preprint arXiv:2403.12570*, 2024.

Jonathan Crabbé and Mihaela van der Schaar. Dynamask: Explaining time series predictions with dynamic masks. In *International Conference on Machine Learning*, 2021.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, and Jan Leike. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, 2025.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in Neural Information Processing Systems*, 2019.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, 2018.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

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

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

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, and János Kramar. Improving dictionary learning with gated sparse autoencoders. In *International Conference on Learning Representations*, 2025.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

Zhiyuan Wang, Xovee Xu, Weifeng Zhang, Goce Trajcevski, Ting Zhong, and Fan Zhou. Learning latent seasonal-trend representations for time series forecasting. In *Advances in Neural Information Processing Systems*, 2022.

A CASE STUDY: WHAT DOES A TSAE FEATURE REPRESENT?

Aggregate diagnostics establish fidelity and code-level behavior, but they do not by themselves show what an individual atom represents. We therefore conduct an auditable case study using the formal TSAE PatchTST checkpoints and test hidden-state caches. The protocol treats a private atom as a variable-specific pair (variable index, local atom index), rather than merging the same local coordinate across private dictionaries, and combines four complementary checks: repeated raw-patch patterns, variable scope, physical strength alignment, and forecast sensitivity.

Repeated patterns and shared/private scope. Figure 4 displays two shared and two private atoms from the formal ETTh1 run. For each atom, positive examples are chosen from distinct raw intervals with a minimum separation of one patch; red controls are also non-overlapping and have negligible activation. The shared atoms recur across several variables, whereas each private atom is evaluated only on its routed variable. Variable scope for private atoms is architectural, while recurrence and physical alignment are checked empirically.

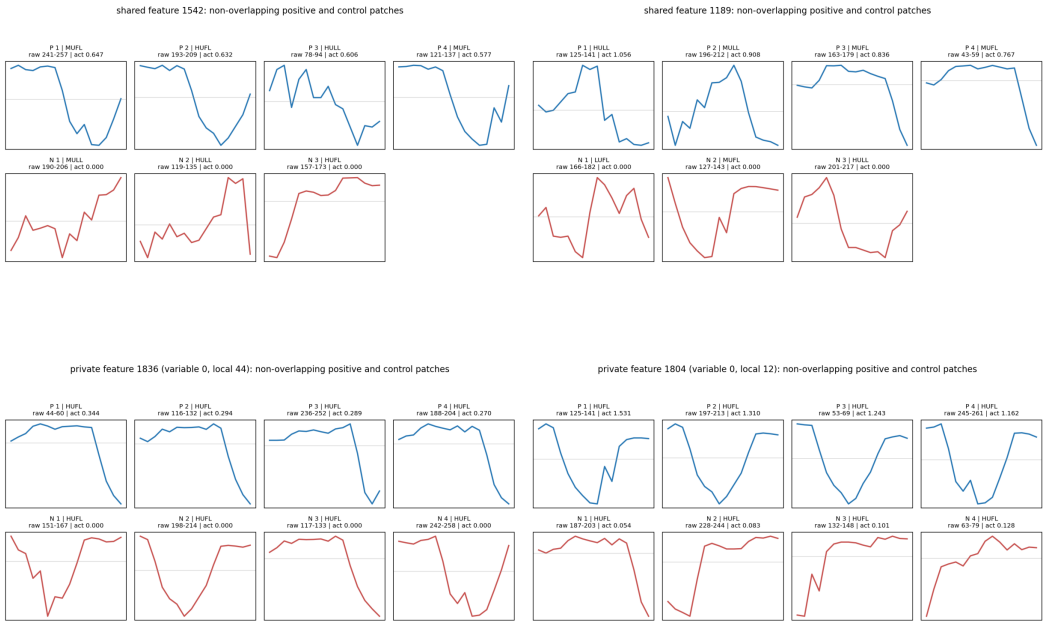


Figure 4: Multi-feature case atlas from the formal PatchTST–ETTh1 run. Blue curves are non-overlapping high-activation patches and red curves are non-overlapping low-activation controls. The first row shows shared atoms that recur across variables; the second row shows private atoms, each restricted to its routed variable. The panel is illustrative; the full pre-recorded 12-atom atlas and selection manifest are stored in the artifact. In-panel annotations are small and are best viewed zoomed in the electronic PDF.

Strength, continuity, and forecast relevance. Figure 5(a) bins a raw physical metric into equal-frequency intervals and plots the mean activation of four selected atoms. The displayed atom–metric pairs were selected by their absolute correlation from the pre-specified slope, volatility, range, curvature, and spike descriptors, so the curves are descriptive illustrations rather than held-out significance tests. The two displayed shared atoms track absolute slope ($r = 0.74$ and 0.72), while the two private atoms track the volatility of their routed variable ($r = 0.60$ for both).

Temporal coherence is not inferred from these examples. Instead, after retaining exactly the top 24 atoms per token for both TSAE and L1 SAE, the mean adjacent-patch Jaccard similarity is higher for TSAE on all three datasets (Table 5). This evaluation-only readout removes unequal support size as an explanation for the comparison.

Figure 5(b) tests forecast relevance by zeroing one atom in the SAE reconstruction and decoding through the frozen forecasting backbone. Each physically aligned atom is paired with a lower-correlation control whose activation rate is within a factor of 1.4. Removing the selected shared atoms produces larger mean forecast shifts than their controls on all three datasets (ratios 2.04, 2.05, and 2.40). Private-atom evidence is strongest on ETTh1 (ratio 2.97) and weaker on ETTh2/ETTm1 (ratios 1.06 and 1.30). Thus the case study supports forecast relevance for selected atoms, but does not claim that every private atom has an equally large downstream effect or that the intervention identifies external physical causality.

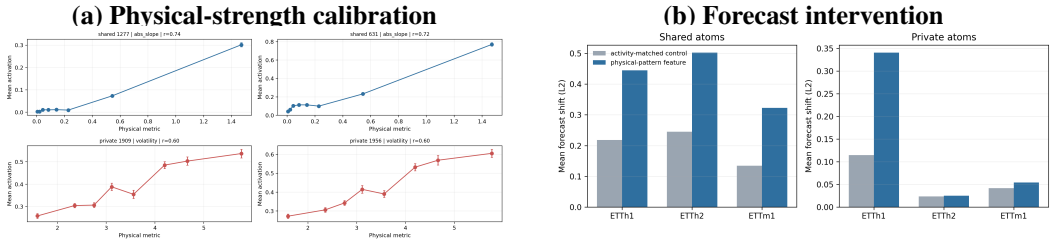


Figure 5: Feature-level evidence beyond reconstruction. (a) ETTh1 activation as a function of the associated raw physical metric; bars show 95% confidence intervals within equal-frequency bins. (b) Mean forecast shift after deleting a selected physical-pattern feature (blue) or an activation-rate-matched, lower-correlation control (gray), while the PatchTST backbone remains frozen. The intervention measures sensitivity within the model pathway, not a causal effect in the data-generating system.

Dataset	L1 SAE Jaccard	TSAE Jaccard	Difference
ETTh1	0.184	0.257	+0.073
ETTh2	0.456	0.676	+0.220
ETTm1	0.424	0.540	+0.116

Table 5: Adjacent-patch support stability at an identical Top-24 feature budget per token. Values are descriptive means over adjacent test-patch pairs for the seed-2024 formal checkpoints.

Audit trail. The case study uses the seed-2024 formal TSAE checkpoint for each ETT dataset and the corresponding recorded test hidden-state cache; it is a feature-level audit, not an additional three-seed method comparison. For the atlas we select eight shared and four private atoms, require four positive examples from distinct samples, and reject any pair of raw intervals that overlap or are separated by fewer than 16 time steps; private controls are drawn only from the owner variable. The artifact stores every selected interval, rejection reason, activation, checkpoint hash, and cache hash.

For the temporal readout, we rank each method's features within each token and retain the same top-24 support before computing adjacent-patch Jaccard similarity. This is an evaluation-only normalization: it neither changes the trained dictionaries nor estimates a new reconstruction metric, and Table 5 should not be compared directly to raw transition rates, whose scale also depends on dictionary width.

For strength calibration, raw patches are assigned to equal-frequency bins of the selected physical descriptor and we report the mean feature activation and within-bin 95% confidence interval. Shared atoms are assessed over all variables; a private atom is assessed only over its routed variable, because zero values outside that domain are structural rather than negative evidence.

For forecast interventions, a single selected atom is set to zero in the SAE code, its reconstruction is decoded by the frozen PatchTST model, and the forecast shift is measured relative to the unablated SAE reconstruction. Each selected atom is paired with a lower physical-correlation control whose

activation rate differs by at most a factor of 1.4. The artifact contains 12 independent top-activation samples for each signal/control atom, per-sample forecast shifts and MSE changes, pair identities, and manifests. This establishes sensitivity along the frozen model pathway; it does not turn the selected motif into a causal variable of the data-generating process.

B Related Work

Time-series explainability. Time-series explainability has traditionally focused on attributing predictions to input variables, timesteps, or subsequences. Saliency methods (Sundararajan et al., 2017), LIME-style local approximations (Ribeiro et al., 2016), dynamic masks (Crabbé & van der Schaar, 2021), Shapley-value methods such as TimeSHAP (Bento et al., 2021), occlusion, and counterfactual explanations identify which parts of an input sequence affect a forecast. Sanity checks for saliency (Adebayo et al., 2018) and attribution benchmarks (Hooker et al., 2019) show that such explanations can be fragile. More importantly for this paper, input-level attribution answers a different question from hidden-feature interpretation: it can locate influential inputs, but it does not test whether an internal sparse feature is reconstructed, semantically coherent, physically calibrated, or forecast-sensitive.

Sparse autoencoders for representation interpretability. Sparse autoencoders have recently become a central tool for interpreting hidden representations in large language models. SAE work studies reconstruction–sparsity tradeoffs (Gao et al., 2025), feature splitting and monosemanticity (Bricken et al., 2023; Bussmann et al., 2024; Lieberum et al., 2023), shrinkage and gated or JumpReLU variants (Rajamanoharan et al., 2025; 2024), and downstream interventions. Related concept-based methods such as TCAV (Kim et al., 2018) and sparse probing (Gurnee et al., 2023) test whether hidden directions align with human-interpretable concepts. These works motivate our use of sparse hidden features, but their standard evidence—reconstruction, sparsity, top activations, and text-domain interventions—does not directly address variable-indexed and physically scaled time-series representations.

SAE-based time-series explanations. Recent work has begun to adapt SAE-style explanations to time-series models, but the role of forecasting hidden-state structure remains underexplored. Forecasting work such as LaST separates latent seasonal and trend components to improve accuracy (Wang et al., 2022). It does not audit sparse hidden features for cross-variable structure, temporal stability, or calibration. TimeSAE is the closest time-series-specific sparse explanation framework to our work (Oublal et al., 2026): it learns sparse concepts to explain the predictions of a black-box time-series model, emphasizing explanation faithfulness under intervention, distribution shift, and compositional consistency.

In contrast, TSAE learns structured sparse features from hidden states extracted from a frozen forecasting backbone, rather than learning a sparse explanation of a black-box predictor's input–prediction behavior. The two perspectives are complementary: TimeSAE asks whether sparse concepts faithfully explain a predictor's behavior, whereas TSAE asks how internal forecasting features should be structured and audited for fidelity, temporal stability, and physical calibration.

Evaluation of time-series sparse features. Prior SAE evaluation practices examine reconstruction–sparsity tradeoffs, activation patterns, and downstream effects (Gao et al., 2025; Bricken et al., 2023). These diagnostics do not, by themselves, directly assess variable structure, temporal stability, or calibration to time-series signals. TSAE specializes SAE design through shared/private dictionaries, magnitude-preserving gates, and conditional temporal continuity. TSEval extends sparse-feature evaluation with reconstruction, semantic/channel purity, physical calibration, forecast intervention sensitivity, hidden controls, and input-level baselines, so that each interpretability claim is tied to a separate diagnostic rather than inferred from reconstruction alone.

C Additional Experimental Details

C.1 Scope and Organization

This appendix documents parameter selection, sensitivity analyses, per-dataset active-matching diagnostics, cross-backbone extensions, and reproducibility information. Section C.2 describes how hyperparameters were chosen and reports the sweeps that justify those choices. Section C.3 reports per-dataset active-matched achievement. Section C.4 extends the comparison to iTransformer and DLinear backbones and reports multi-backbone component ablations. Section C.5 provides reproducibility details. Appendix A records the feature-level case-study protocol and its activity-controlled temporal readout.

C.2 Configuration & Sensitivity

Selection procedure

For each dataset and each scale-sensitive method, we ran a validation sweep over the sparsity-related parameter, recorded the mean active features per token, and selected the setting whose validation activity was closest to 256. Configurations with mean activity in $[192, 320]$ were admissible; ties were broken by validation reconstruction MSE. TopK used an exact activity budget ($k = 256$ for the ETT datasets and $k = 320$ for Weather). The selected setting was then fixed for three-seed test-split retraining.

Selected operating points

Dataset	Method	Selected setting	Test Active
ETTh1	TSAE	shared $\lambda = 6.0$, private $\lambda = 0.30$	231.4±2.7
	L1	$\lambda = 2.0$	252.8±10.3
	Gated	$\lambda = 6.0$	244.4±7.4
	TopK	$k = 320$	264.9±5.7
	JumpReLU	λ=10^{-4}, threshold=0.10	269.8±11.4
ETTh2	TSAE	shared $\lambda = 3.0$, private $\lambda = 0.15$	250.9±13.6
	L1	$\lambda = 1.5$	195.2±11.4
	Gated	$\lambda = 5.0$	212.7±6.0
	TopK	$k = 256$	218.3±1.0
	JumpReLU	λ=10^{-4}, threshold=0.08	311.0±39.9
ETTm1	TSAE	shared $\lambda = 5.0$, private $\lambda = 0.25$	204.9±11.5
	L1	$\lambda = 1.5$	258.0±11.9
	Gated	$\lambda = 5.0$	246.9±5.8
	TopK	$k = 256$	223.7±2.4
	JumpReLU	λ=10^{-4}, threshold=0.09	269.9±12.8
Weather	TSAE	shared $\lambda = 1.8$, private $\lambda = 0.09$	216.7±3.2
	L1	$\lambda = 1.0$	187.8±16.3
	Gated	$\lambda = 2.0$	285.1±8.8
	TopK	$k = 320$	261.1±14.4
	JumpReLU	λ=10^{-4}, threshold=0.08	236.0±0.7

Table 6: Validation-selected operating points with test-split active counts. TSAE runs use $\lambda_{\text{temp}} = 0.5$, $\lambda_{\text{fc}} = 0.1$, and $\lambda_{\text{balance}} = 0.003$.

All ETT test means in Table 6 fall within the prespecified activity window. On Weather, the L1 test mean drifts slightly below the lower boundary (187.8), while the remaining methods remain in-window. This protocol controls realized activity per token; it does not equalize the total parameter count of TSAE's routed dictionaries and flat baselines.

CALIBRATION SELECTION CURVES

Figure 6 shows how the measured active count responds to the sparsity parameter for each method. The gray band marks the [192,320] admissible window. For TSAE, the active count decreases monotonically as the shared sparsity coefficient increases; the selected operating points are those closest to 256. For L1 and Gated the relationship is similarly monotonic. JumpReLU and TopK have narrower tuning ranges.

TEMPORAL COEFFICIENT SENSITIVITY

We swept λ_{temp} on PatchTST–ETTh1 (single seed, $\lambda_{\text{shared}} = 5.0$, $\lambda_{\text{priv}} = 0.2$). Table 7 reports the results. Below 0.01, the transition metric is effectively unchanged. At 0.5 it decreases by about 7% relative to zero while the realized activity remains near the target. The single-seed curve continues to change at 1.0 together with activity, so it is a sensitivity diagnostic rather than an independent optimality claim. The formal comparison uses the validation-selected coefficient 0.5.

λ_{temp}	Active	Recon.↓	NFRE↓	Trans.↓
0	280	12.3	5.1	0.0908
0.0001	280	12.3	5.1	0.0909
0.001	280	12.3	5.1	0.0908
0.01	278	12.3	5.1	0.0901
0.05	276	12.2	5.1	0.0903
0.1	273	12.2	5.1	0.0895
0.5	**255**	**11.7**	**4.9**	**0.0844**
1.0	236	11.0	4.5	0.0787
0.5 (3-seed)	268±11	11.7±0.1	4.9±0.1	0.086±0.003

Table 7: Temporal coefficient sweep (single seed unless noted). The bold row is the validation-selected formal coefficient. Recon. and NFRE are $\times 10^3$.

PRIVATE SPARSITY SENSITIVITY

We swept TSAE's private λ on PatchTST–ETTh1 (single seed, $\lambda_{\text{shared}} = 5.0$). A larger coefficient ($\lambda_{\text{priv}} \approx 1.0$) produced 13.8% dead features. Lowering the coefficient to 0.05–0.2 reduces the dead ratio in this fixed-setting diagnostic while improving reconstruction and NFRE. The sweep is illustrative: final formal settings are selected separately for each dataset under the active-matched protocol.

λ_{priv}	Active	Recon.↓	NFRE↓	Dead%
0.02	500	10.7	4.4	6.2
0.05	489	**9.4**	**3.8**	2.3
0.1	456	9.5	3.9	0.6
0.2	**416**	10.2	4.2	1.0
0.5	392	11.0	4.4	7.0
1.0	386	11.3	4.5	13.8

Table 8: Private sparsity sensitivity at a fixed shared coefficient (single seed). Bold marks the illustrative low-dead-feature range; formal comparisons use dataset-specific validation-selected values.

C.3 PER-DATASET ACTIVE-MATCHED ACHIEVEMENT

The main-text table compresses the formal comparison. The four tables below expand it per dataset, showing whether each method's mean test-split active count falls within the prespecified [192,320] window. The "In window" column provides a direct check of the active-matched protocol.

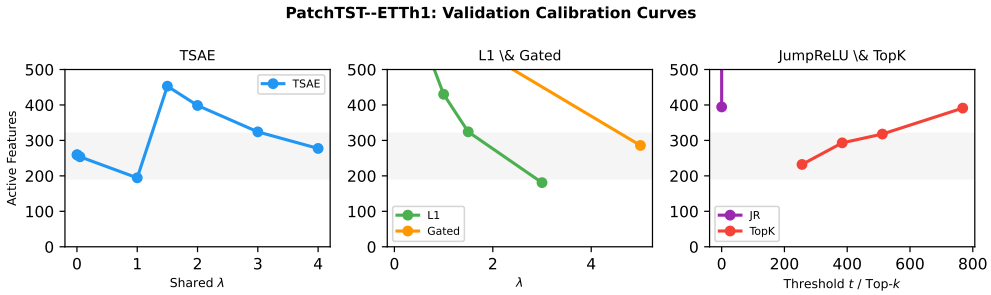


Figure 6: Validation calibration curves on PatchTST–ETTh1. Horizontal axis: sparsity parameter; vertical axis: measured active features per token. Gray band: admissible [192,320] window.

Method	Active	In 192–320?	Recon.↓	NFRE↓	Dead%	Trans.↓
TSAE	231±3	✓	**13.4**	**5.10**	8.0	**0.073**
L1	253±10	✓	13.6	5.33	**0.0**	0.181
Gated	244±7	✓	16.4	6.39	**0.0**	0.144
TopK	265±6	✓	30.5	10.84	0.1	0.143
JumpReLU	270±11	✓	20.7	8.09	**0.0**	0.156

Table 9: ETTh1: all five methods within the [192,320] window. TSAE has the best Recon, NFRE, and Transition rate.

Method	Active	In 192–320?	Recon.↓	NFRE↓	Dead%	Trans.↓
TSAE	251±14	✓	**6.04**	**2.70**	4.3	**0.050**
L1	195±11	✓	7.99	3.66	2.5	0.106
Gated	213±6	✓	8.19	3.76	10.3	0.055
TopK	218±1	✓	9.88	4.56	14.3	0.063
JumpReLU	311±40	✓	10.9	4.79	**4.1**	0.064

Table 10: ETTh2: all five methods are within the [192,320] window. TSAE is the best on Recon, NFRE, and Transition.

ETTh1

ETTh2

ETTm1

Method	Active	In 192–320?	Recon.↓	NFRE↓	Dead%	Trans.↓
TSAE	205±12	✓	**10.4**	**5.00**	3.1	**0.047**
L1	258±12	✓	11.4	5.50	**0.0**	0.135
Gated	247±6	✓	13.5	6.58	0.0	0.089
TopK	224±2	✓	19.9	8.02	0.6	0.083
JumpReLU	270±13	✓	17.1	8.63	**0.0**	0.092

Table 11: ETTm1: all five methods within the [192,320] window. TSAE has the best Recon, NFRE, and Transition rate.

Weather

This breakdown shows that validation-based active matching is generally well-controlled across the datasets. On Weather, L1's selected validation point is in-window, while its three-seed test mean is 188; the other four methods remain in-window. TSAE retains the best fidelity and temporal-stability metrics at a comparable realized per-token activity level.

Method	Active	In 192–320?	Recon.↓	NFRE↓	Dead%	Trans.↓
TSAE	217±3	✓	**1.70**	**2.52**	19.5	**0.019**
L1	188±16	×	<u>1.85</u>	3.37	**0.0**	0.055
Gated	285±9	✓	2.10	<u>2.84</u>	<u>0.0</u>	0.046
TopK	261±14	✓	3.84	4.64	7.3	0.038
JumpReLU	236±1	✓	4.49	6.21	0.1	<u>0.031</u>

Table 12: Weather: the validation-selected settings yield in-window test means for TSAE, Gated, TopK, and JumpReLU. L1 is slightly below the boundary at 188, despite its validation point lying in-window. TSAE has the best Recon, NFRE, and Transition.

C.4 Cross-Backbone Extensions

iTransformer–ETTh1 formal comparison

We ran the same active-matched protocol on frozen iTransformer representations from ETTh1, using a 1024-feature per-token budget and the same 20-epoch, 30,000/15,000-token budget. Table 13 reports three-seed test-split means.

Method	Active	Recon.↓	NFRE↓
TSAE	253±3	**14.5**	**3.01**
L1	282±1	18.1	4.14
Gated	316±1	17.3	3.20
TopK	212±0	30.1	5.24
JumpReLU	269±3	18.8	3.44

Table 13: iTransformer–ETTh1 active-matched results (three-seed means). Recon. and NFRE are $\times 10^3$.

TSAE yields the lowest reconstruction and NFRE on the iTransformer backbone, consistent with the PatchTST headline result.

DLinear–ETTh1 formal comparison

Table 14 reports the same protocol on frozen DLinear representations from ETTh1.

Method	Active	Recon.↓	NFRE↓
TSAE	215±10	9.09	7.91
L1	250±1	9.17	8.74
Gated	314±1	**7.71**	**6.88**
TopK	256±0	9.90	8.27
JumpReLU	266±4	7.99	7.02

Table 14: DLinear–ETTh1 active-matched results (three-seed means). Recon. and NFRE are $\times 10^3$.

On DLinear, Gated SAE has the lowest Reconstruction and NFRE; TSAE is third on NFRE and second on Reconstruction. The fidelity advantage does not automatically transfer to every architecture—backbone representation geometry matters.

Multi-backbone component ablation

Tables 15–18 report the component ablation across four backbones on three datasets. Each table shows three-seed means with bold marking the best value per column per dataset. The ablation uses a fixed $\lambda_{\text{shared}} = 5.0$, $\lambda_{\text{priv}} = 0.2$, $\lambda_{\text{temp}} = 0.5$ configuration for TSAE. The No temporal variant is included only for PatchTST; all three other backbones use representation types without ordered temporal adjacency.

Data	Variant	Active	Recon.↓	NFRE↓	Dead%	Trans.↓
ETTh1	Full	268	**11.7**	**4.9**	6.3	**0.086**
	No routing	252	13.4	5.5	**0.0**	0.145
	No gating	108	18.1	7.9	**0.0**	0.083
	No temporal	279	12.3	5.1	6.3	0.091
ETTh2	Full	203	**2.2**	**3.4**	10.9	**0.036**
	No routing	178	2.4	3.6	13.1	0.050
	No gating	46	3.7	5.7	23.3	0.027
	No temporal	234	2.8	4.2	13.3	0.038
ETTm1	Full	218	**9.1**	**5.2**	3.8	**0.049**
	No routing	192	9.8	5.7	0.1	0.071
	No gating	79	14.1	8.5	**0.0**	0.047
	No temporal	261	10.3	5.8	6.2	0.052

Table 15: PatchTST component ablation (three-seed means). Bold marks the best per column per dataset.

Data	Variant	Active	Recon.↓	NFRE↓	Dead%
ETTh1	Full	399	**13.7**	**3.3**	0.0
	No routing	309	23.3	4.3	0.0
	No gating	188	34.7	7.8	0.0
ETTh2	Full	360	**22.9**	**1.1**	0.0
	No routing	284	31.6	1.4	0.0
	No gating	187	45.7	1.9	0.0
ETTm1	Full	383	**14.2**	**4.2**	0.0
	No routing	290	23.1	5.8	0.0
	No gating	154	34.4	10.1	0.0

Table 16: iTransformer component ablation. Transition rate not applicable.

Data	Variant	Active	Recon.↓	NFRE↓	Dead%
ETTh1	Full	219	**21.3**	**9.8**	2.4
	No routing	165	25.7	11.6	**0.0**
	No gating	121	43.5	21.9	**0.0**
ETTh2	Full	153	**7.9**	**7.3**	5.9
	No routing	117	8.0	7.2	**0.0**
	No gating	87	15.0	13.2	**0.0**
ETTm1	Full	139	**19.5**	**13.1**	4.9
	No routing	104	21.6	15.0	**0.0**
	No gating	63	35.5	24.3	**0.0**

Table 17: DLinear component ablation.

Data	Variant	Active	Recon.↓	NFRE↓	Dead%
ETTh1	Full	322	**13.6**	**3.5**	3.2
	No routing	281	18.7	5.0	**0.0**
	No gating	126	38.1	12.8	**0.0**
ETTh2	Full	287	**4.2**	**4.6**	8.3
	No routing	221	5.6	6.1	**8.7**
	No gating	74	12.7	15.8	**0.0**
ETTm1	Full	289	**8.8**	**5.0**	8.6
	No routing	209	10.9	6.4	**3.3**
	No gating	83	28.6	18.1	**0.0**

Table 18: Transformer component ablation.

Across this fixed-configuration matrix, removing routing generally degrades reconstruction and removing gating lowers realized activity substantially. The direction of these component effects is

observed across several representation geometries, but the variants are not activity-matched; these tables are therefore supporting component diagnostics rather than a universal cross-backbone performance comparison.

C.5 REPRODUCIBILITY

All experiments use frozen forecasting backbones trained for 10 epochs with seed 2024. SAE training uses 20 epochs, batch size 1024, learning rate 10^{-3} (Adam), on at most 30,000 training tokens and 15,000 test tokens. For the PatchTST headline comparison, TSAE requests $K_{\mathrm{sh}} = 1792$ shared and $K_{\mathrm{priv}} = 256$ private features per variable. A resolver enforces $K_{\mathrm{global}} \leq 4096$, so the realized shared/private allocation can depend on the variable count; flat baselines use a single 2048-wide dictionary. PatchTST TSAE runs additionally use $\lambda_{\mathrm{fc}} = 0.1$ for two refinement epochs with at most 80 batches per epoch; baselines do not use this TSAE-specific stage. The iTransformer and DLinear extensions use their separately reported 1024-feature budgets. Formal results are averaged over three independent SAE seeds $\{2024, 2025, 2026\}$. Experiments were run on NVIDIA RTX 4090 (24 GB) and RTX 4080 SUPER (32 GB) with PyTorch 2.10+cu128 and CUDA 12.8. Complete run-level results, calibration manifests, and figure-generation scripts are included in the released artifact.

REPRODUCIBILITY CHECKLIST

General paper information.

- Includes a conceptual outline and/or pseudocode description of AI methods introduced: **yes**. Section "TSAE: Structured Sparse Autoencoder" defines the shared/private dictionaries, gated encoder, temporal regularizer, and training objective; Section "TSEVAL: Time-Series SAE Evaluation Protocol" gives the five-stage diagnostic procedure and the six diagnostic axes.
- Clearly delineates opinions, hypotheses, and speculation from objective facts and results: **yes**. Interpretation rules, evidence hierarchy, limitations, and the supporting-evidence labels separate measured results from diagnostic recommendations.
- Provides pedagogical references for readers to gain background necessary to replicate the paper: **yes**. Related work covers SAE evaluation, concept testing, time-series explanation, and forecasting architectures.

Theoretical contributions. Does this paper make theoretical contributions? **no**. The paper is primarily empirical and methodological. The argument that the six TSEVAL diagnostic axes are non-redundant is presented as a conceptual motivation within the protocol section rather than as formal propositions.

- Formal assumptions and statements are provided: **NA**.
- Proof sketches or justifications are provided: **NA**.
- Theoretical claims are connected to experiments: **NA**.
- Code for theoretical claims: **NA**.

Datasets. Does this paper rely on one or more datasets? **yes**.

- Motivation for selected datasets: **yes**. Section "Setup and Evidence Hierarchy" motivates ETT and Weather as public long-horizon forecasting benchmarks with different variable counts and sampling frequencies.
- Novel datasets introduced: **NA**. The paper introduces no new dataset.
- Existing datasets are accompanied by appropriate citations: **yes**. The paper cites prior forecasting work using ETT/Weather, and the artifact README lists dataset sources.
- Existing datasets are publicly available: **yes**. The artifact README provides public dataset locations.
- Non-public datasets: **NA**. No private dataset is used.

Computational experiments. Does this paper include computational experiments? **yes**.

- Number and range of hyperparameter values tried during development, plus final selection criterion: **yes**. The main text lists headline settings, and the artifact reproducibility appendix records final settings, development-run families, and the selection rule used to choose the reported matrix.

- Pre-processing code is included in the appendix/artifact: **yes**. Data loading, hidden-cache construction, input baselines, and aggregation entry points are included in the released scripts.
- Source code required for conducting and analyzing experiments is included in a code appendix/artifact: **yes**. The artifact contains benchmark aggregation, all-model adapter smoke tests, and validation-sweep entry points; the repository contains the analysis implementation.
- Source code will be publicly available upon publication with a research-use license: **yes**. The intended release is the submitted repository/artifact package.
- Source code implementing new methods has comments detailing implementation links to the paper: **yes**. The adapter registry and SAE-analysis entry point include paper-to-code mapping comments, and the reproducibility appendix maps claims to scripts and artifacts.
- Randomness and seed handling are described sufficiently to replicate results: **yes**. Released scripts expose separate backbone and SAE initialization variables, and the appendix reports every run-level value.
- Computing infrastructure is specified: **yes**. The appendix records the final GPU and software environment, and the artifact contains the environment-capture output.
- Evaluation metrics are formally described and motivated: **yes**. Reconstruction, semantic/channel purity, physical calibration, FIS, hidden-direct controls, and input baselines are defined in the protocol section.
- Number of algorithm runs used to compute each reported result is stated: **yes**. Headline results use three independent SAE runs; single-run evidence is explicitly marked as supporting.
- Analysis goes beyond single-dimensional summaries: **yes**. The paper reports mean and standard deviation, paired run directions, multi-axis diagnostics, and failure-mode analyses.
- Significance of improvements/decreases is judged with appropriate statistical checks: **NA**. With three runs, we make no inferential significance claim; uncertainty is reported descriptively and all run-level values are exposed.
- Final hyperparameters used for each model/algorithm are listed: **yes**. Main-text settings, scripts, per-run configuration files, and the artifact reproducibility appendix specify the headline configurations.