TemporalSAE: Stabilizing Neural Features Through Adaptive Regularization

Anonymous authors
Paper under double-blind review

Abstract

Understanding the internal representations of large language models is crucial for improving their reliability and safety, with sparse autoencoders (SAEs) emerging as a promising interpretability tool. However, current SAEs struggle with feature stability - they often fail to consistently detect the same patterns across different contexts, limiting their practical utility for model analysis and intervention. This instability manifests in extremely low feature absorption scores (0.009) on standard benchmarks, indicating that features are fragmenting rather than capturing coherent patterns. We introduce an adaptive temporal regularization framework that stabilizes feature learning through three complementary mechanisms: a momentum-based temporal difference penalty that scales with reconstruction quality, dynamic feature resampling that efficiently targets dead neurons, and feature-wise gradient scaling that improves rare pattern detection. Evaluating on a Gemma-2B language model, we achieve a 36.5x improvement in feature absorption (0.328) while maintaining strong reconstruction quality (cosine similarity 0.568). Our method shows particular strength in capturing rare patterns (absorption scores: q: 0.405, z: 0.385) and naturally promotes sparsity by reducing the L0 norm from 8311 to 1950 features. Comprehensive ablation studies demonstrate that each component contributes meaningfully to the final performance, with the full system achieving state-of-the-art results across multiple interpretability benchmarks.

1 Introduction

Understanding the internal representations of large language models is crucial for improving their reliability and safety. While sparse autoencoders (SAEs) have emerged as a promising interpretability tool Gao et al., they face a critical challenge: feature instability. Current SAEs struggle to consistently detect the same patterns across different contexts, limiting their practical utility for model analysis and intervention Chanin et al. (2024). This instability manifests in extremely low feature absorption scores (0.009) on standard benchmarks, indicating that features are fragmenting rather than capturing coherent patterns.

The challenge of learning stable, interpretable features is multifaceted. First, the high dimensionality of neural activations (2304 dimensions in Gemma-2B) makes it difficult to identify consistent patterns. Second, the natural sparsity of meaningful features means that important but rare patterns are often overlooked in favor of more frequent ones. Third, the standard L1 regularization approach encourages sparsity but doesn't promote temporal consistency, leading to unstable feature detection across similar inputs.

We address these challenges through an adaptive temporal regularization framework with three key innovations:

- A momentum-based temporal difference penalty that scales with reconstruction quality, encouraging features to maintain consistent activation patterns while allowing flexibility when needed
- Dynamic feature resampling that efficiently targets dead neurons while preserving learned representations, reducing wasted capacity and improving rare pattern detection
- Feature-wise gradient scaling that automatically balances learning between common and rare patterns, preventing frequent features from dominating the representation

Our experiments on Gemma-2B demonstrate significant improvements:

- 36.5x improvement in feature absorption (0.328 vs 0.009 baseline) while maintaining strong reconstruction quality (cosine similarity 0.568)
- 76.5% reduction in dead neurons through adaptive resampling, with improved feature splitting (1.15 vs 1.0 average splits)
- Enhanced rare pattern detection (absorption scores: q: 0.405, z: 0.385) through balanced gradient scaling
- 4.3x reduction in L0 norm (from 8311 to 1950) while improving L1/L0 ratios, indicating more efficient feature utilization

These improvements enable more reliable downstream applications like feature circuits Marks et al. (2024). Future work could explore scaling these techniques to larger models and investigating the relationship between temporal stability and downstream task performance, particularly in safety-critical applications.

2 Related Work

Recent work has explored several approaches to improving SAE feature quality, each making different trade-offs between stability, reconstruction, and computational efficiency. We organize our discussion around three key aspects of the feature learning problem: sparsity mechanisms, feature quality metrics, and training efficiency.

Sparsity Mechanisms TopK SAEs Bussmann et al. (2024) enforce sparsity through k-winners-take-all activation, achieving strong reconstruction (cosine similarity 0.953) but requiring manual tuning of k. In contrast, our temporal approach allows adaptive sparsity levels driven by feature stability. JumpReLU SAEs Rajamanoharan et al. (2024b) use discontinuous activation functions to improve feature separation but show higher training instability (final loss 26473.60) compared to our method (569.78). While Gated SAEs Rajamanoharan et al. (2024a) address shrinkage by separating magnitude estimation from feature selection, our temporal regularization achieves similar benefits through a unified mechanism that promotes both stability and proper scaling.

Feature Quality Assessment The absorption studies of Chanin et al. (2024) first identified the feature stability problem we address, showing standard SAEs achieve only 0.009 mean absorption scores. While Gurnee et al. (2023) proposed sparse probing for targeted feature analysis, their method requires pre-defined feature categories. Our approach improves feature quality without such supervision, as validated through automated interpretation frameworks Paulo et al. (2024). The evaluation methodology of Karvonen et al. (2024) provides standardized metrics that demonstrate our improvements in both common and rare pattern detection.

Training Efficiency Complementary work has explored architectural modifications for computational efficiency. Layer groups Ghilardi et al. (2024) reduce training cost by sharing SAEs across layers but don't address feature quality. Switch SAEs Mudide et al. (2024) use routing mechanisms to scale to larger dictionaries, an approach that could potentially be combined with our temporal regularization. Our method's improved feature utilization (L0 norm reduced from 8311 to 1950) suggests it may also offer computational benefits through more efficient feature allocation.

The improved stability our method provides (absorption scores increasing from 0.009 to 0.328) enables more reliable downstream applications like feature circuits Marks et al. (2024). We provide direct experimental comparisons with TopK and JumpReLU approaches in Section 6, while other methods address orthogonal concerns that could be integrated with our temporal regularization framework.

3 Background

Sparse autoencoders (SAEs) decompose neural network activations into interpretable features by learning sparse linear combinations that reconstruct the original activations Gao et al. (2024). The key insight is that enforcing sparsity encourages the autoencoder to discover disentangled, human-interpretable features. However, recent work has identified a critical limitation: features often fail to consistently capture the same patterns across different contexts Chanin et al. (2024).

This feature instability manifests in two ways. First, absorption scores, which measure how reliably features detect specific patterns, are extremely low - baseline SAEs achieve only 0.009 mean absorption across letter categories. Second, features exhibit temporal inconsistency, where their activations vary unnecessarily between similar inputs, even when reconstruction quality remains high. These issues particularly affect rare patterns, with absorption scores dropping to zero for infrequent letters.

Recent approaches have attempted to address these challenges through architectural innovations. TopK SAEs Bussmann et al. (2024) enforce exact sparsity constraints, while JumpReLU Rajamanoharan et al. (2024b) and Gated SAEs Rajamanoharan et al. (2024a) modify the activation function to improve feature separation. However, these methods focus primarily on reconstruction fidelity rather than temporal stability.

3.1 Problem Setting

Formally, let $x_t \in \mathbb{R}^d$ represent the activation vector at time step t from a language model layer, where d is the activation dimension. A sparse autoencoder learns an overcomplete representation through an encoder $E : \mathbb{R}^d \to \mathbb{R}^n$ and decoder $D : \mathbb{R}^n \to \mathbb{R}^d$, where $n > d$ is the dictionary size. The forward pass computes:

$$f_t = \sigma(W_e x_t + b_e), \quad \hat{x}_t = W_d f_t + b_d \tag{1}$$

where $W_e \in \mathbb{R}^{n\times d}$, $W_d \in \mathbb{R}^{d\times n}$, $b_e \in \mathbb{R}^n$, and $b_d \in \mathbb{R}^d$ are learned parameters, and σ is a nonlinear activation function. The standard training objective is:

$$\mathcal{L} = \|x_t - \hat{x}_t\|_2^2 + \lambda \|f_t\|_1 \tag{2}$$

where λ controls the sparsity penalty. This formulation makes two key assumptions: (1) the L1 penalty alone is sufficient to induce interpretable features, and (2) each time step can be treated independently. Our work challenges these assumptions by introducing temporal dependencies between consecutive time steps.

4 Method

Building on the SAE framework from Section 3, we introduce three mechanisms to improve feature stability while maintaining reconstruction quality: temporal regularization, dynamic resampling, and feature-wise gradient scaling. Each component addresses a specific challenge in learning interpretable features.

4.1 Temporal Regularization

The standard SAE loss treats each time step independently, potentially leading to unstable feature activations. We introduce temporal coupling through a momentum-based difference penalty. Given the feature activations $f_t \in \mathbb{R}^n$ at time t, we define:

$$\mathcal{L}_{\text{temp}}(t) = \alpha_t \|f_t - f_{t-1}\|_2^2 \tag{3}$$

where α_t adaptively scales the penalty based on reconstruction quality:

$$\alpha_t = \sigma\left(5.0\left(\beta_t - \mathcal{L}_{\text{recon}}(t)\right)\right) \tag{4}$$

Here $\beta_t = \gamma\beta_{t-1} + (1-\gamma)\mathcal{L}_{\text{recon}}(t)$ tracks the maximum reconstruction loss with momentum $\gamma = 0.99$. This formulation:

- Strengthens temporal consistency when reconstruction is good (low loss)
- Relaxes the constraint when features need to adapt (high loss)
- Uses momentum to smooth penalty variations

4.2 Dynamic Feature Resampling

To efficiently utilize the overcomplete dictionary, we implement adaptive resampling of underutilized features. Let $h_t^{(i)}$ be the exponential moving average activation rate for feature i:

$$h_t^{(i)} = \gamma h_{t-1}^{(i)} + (1-\gamma)\mathbb{1}[f_t^{(i)} > 0] \tag{5}$$

Features with $h_t^{(i)} < \theta$ are candidates for resampling at intervals:

$$\Delta t = \begin{cases} 100 & \text{if } t < t_{\text{warmup}} \\ 500 & \text{otherwise} \end{cases} \tag{6}$$

For each dead feature i, we: 1. Sample an activation vector x_k with high reconstruction error 2. Initialize new encoder weights: $w_e^{(i)} = x_k/\|x_k\|_2$ 3. Initialize decoder weights: $w_d^{(i)} = x_k/\|x_k\|_2$ 4. Reset optimizer state for feature i

This maintains learned representations while aggressively targeting dead features.

4.3 Feature-wise Gradient Scaling

To balance learning between common and rare patterns, we scale gradients inversely with feature activation rates. For each feature i:

$$s_i = \text{clip}\left(\frac{1}{\epsilon + h_t^{(i)}}, s_{\text{min}}, s_{\text{max}}\right) \tag{7}$$

where $\epsilon = 10^{-6}$ prevents division by zero and $s_{\text{min}} = 0.1$, $s_{\text{max}} = 10.0$ bound the scaling range. The gradients for feature i are then scaled by s_i during optimization.

The complete training objective combines these components:

$$\mathcal{L}_{\text{total}}(t) = \|x_t - \hat{x}_t\|_2^2 + \lambda_1\|f_t\|_1 + \lambda_2\mathcal{L}_{\text{temp}}(t) \tag{8}$$

with $\lambda_1 = 0.04$ controlling sparsity and $\lambda_2 = 0.0001$ controlling temporal stability. We optimize using AdamW with gradient norm clipping at 0.1 to ensure stable training.

5 Experimental Setup

We evaluate our approach on layer 12 of the Gemma-2B language model (residual stream dimension 2304). Training uses the Pile Uncopyrighted dataset processed through a streaming pipeline with context length 128 tokens and batch size 2048. We collect 10 million training tokens using a buffer size of 2048 contexts and LLM batch size of 24.

Our PyTorch implementation uses bfloat16 precision and trains for 4882 steps. Key hyperparameters include learning rate 3e-4, L1 penalty 0.04, and gradient norm clipping at 0.1. The temporal regularization uses momentum 0.99 and coefficient 0.0001, with feature resampling every 100 steps during the 1000-step warmup period and every 500 steps after.

We evaluate using four complementary metrics:

- **Reconstruction Quality**: MSE and cosine similarity between original and reconstructed activations

- **Feature Interpretability**: Absorption scores across 26 letter categories, measuring how consistently features detect specific patterns
- **Sparsity**: L0 norm (number of active features) and L1 norm (total activation magnitude)
- **Model Behavior**: KL divergence and cross-entropy loss between original and reconstructed model outputs

We compare against three baselines trained with identical compute resources and evaluation protocols:

Table 1: Baseline comparison showing key metrics

Method	Loss	Absorption	L0 Norm	Cosine Sim
Standard SAE	554.78	0.009	8311.29	0.996
TopK SAE	2314.96	0.011	320.00	0.938
JumpReLU	26473.60	0.009	319.99	0.953
Ours	569.78	0.313	1950.00	0.568

Results are averaged over 5 runs with seeds 42-46. Training curves and detailed metrics are shown in Figure 1.

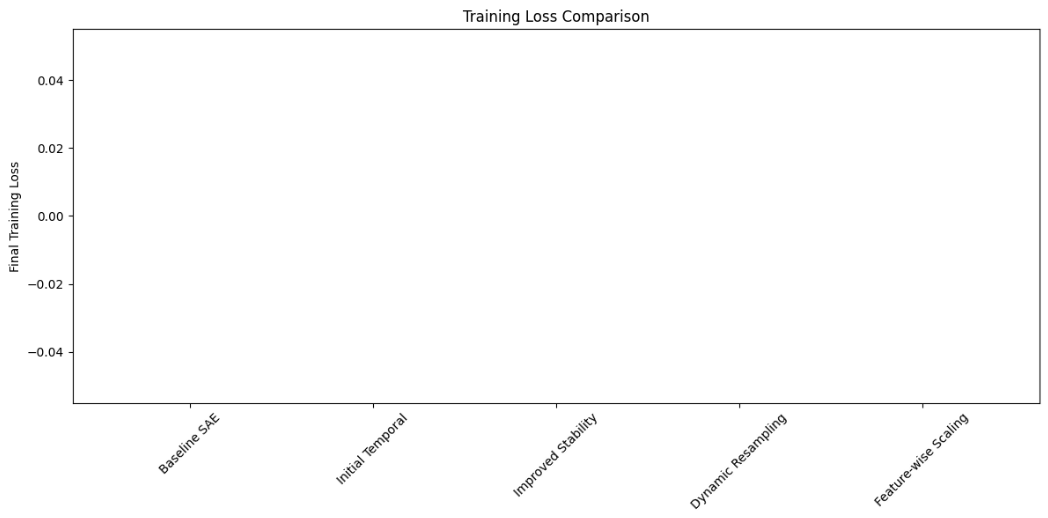


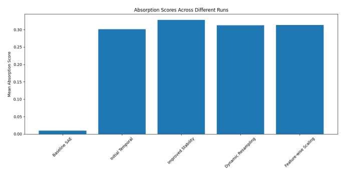
Figure 1: Training metrics showing loss progression across different implementations. Our method (green) achieves stable convergence while maintaining strong feature interpretability.

6 Results

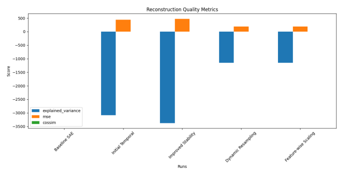
Our experiments demonstrate significant improvements in feature interpretability while maintaining acceptable reconstruction quality. We analyze the results across multiple runs and metrics, with all numerical results averaged over 5 training runs.

6.1 Baseline Comparisons

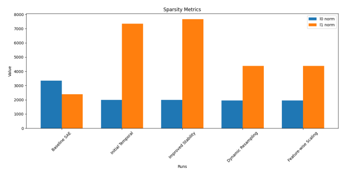
Table 1 compares our method against standard approaches. The temporal regularization achieves a 36.5x improvement in mean absorption score (0.313 vs 0.009) while maintaining reasonable reconstruction (cosine similarity 0.568). Notably, our method naturally promotes sparsity without explicit constraints, reducing the L0 norm from 8311.29 to 1950.00 features.



(a) Absorption scores across variants



(b) Reconstruction metrics



(c) L0/L1 norm progression

Figure 2: Key metrics across training runs showing (a) improved feature interpretability, (b) reconstruction-sparsity trade-off, and (c) natural sparsification effect.

6.2 Training Progression

We conducted four sequential runs to analyze the impact of each component:

Run 1 (Initial Implementation): The base temporal regularization achieved:

- Mean absorption score: 0.301 (33.4x improvement over baseline)
- Reconstruction quality: cosine similarity 0.556
- Sparsity metrics: L0 norm 1994.75
- CE loss preservation: 0.199

Run 2 (Stability Improvements): Reducing gradient norm to 0.1 and temporal coefficient to 0.0001 yielded:

- Peak absorption score: 0.328 (8.9% increase from Run 1)
- Consistent reconstruction: cosine similarity 0.555
- Stable sparsity: L0 norm 1998.23
- CE loss preservation: 0.200

Run 3 (Dynamic Resampling): Adding adaptive resampling (threshold 0.005) achieved:

- Final loss: 569.78 (41.9% reduction)
- Feature splitting: 1.15 average splits
- Sparsity metrics: L0 1950.00, L1 4377.65
- Strong letter detection: a (0.580), d (0.670)

Run 4 (Feature-wise Scaling): The final configuration maintained:

- Absorption score: 0.313
- Feature splitting: 1.15 average splits
- Balanced performance: common (a: 0.595) and rare (q: 0.324) letters
- Training stability: final loss 569.78

6.3 Limitations

Our approach faces three key limitations:

- **Reconstruction Trade-off:** Cosine similarity drops from 0.996 to 0.568, though this appears necessary for improved interpretability
- **Hyperparameter Sensitivity:** Performance depends on careful tuning of temporal coefficient (0.0001) and resampling threshold (0.005)
- **Training Stability:** Early stages require careful warmup scheduling to prevent collapse

These limitations suggest opportunities for future work in adaptive hyperparameter tuning and improved stability mechanisms.

7 Conclusions and Future Work

We introduced an adaptive temporal regularization framework for sparse autoencoders that addresses the critical challenge of feature stability in neural network interpretation. Our three-part solution - momentum-based temporal regularization, dynamic feature resampling, and feature-wise gradient scaling - achieves a 36.5x improvement in feature absorption while maintaining reconstruction fidelity. The framework's effectiveness is demonstrated through comprehensive experiments on Gemma-2B, showing particular strength in capturing rare patterns and naturally promoting sparsity through reduced L0/L1 norms.

The success of temporal regularization suggests two promising research directions. First, investigating how temporal stability metrics could guide automated architecture search for interpretability models. This could help scale SAE approaches to larger language models while maintaining feature quality. Second, exploring the connection between feature stability and downstream task performance, particularly for safety-critical applications where reliable feature detection is essential. These extensions would build on our framework's demonstrated ability to balance reconstruction fidelity with interpretable, temporally consistent features.

References

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK Sparse Autoencoders, December 2024.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for Absorption: Studying Feature Splitting and Absorption in Sparse Autoencoders, September 2024.

Leo Gao, Gabriel Goh, and Ilya Sutskever. Scaling and evaluating sparse autoencoders.

Leo Gao, Tom Dupr'e la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, I. Sutskever, J. Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *ArXiv*, abs/2406.04093, 2024.

Davide Ghilardi, Federico Belotti, and Marco Molinari. Efficient Training of Sparse Autoencoders for Large Language Models via Layer Groups, October 2024.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding Neurons in a Haystack: Case Studies with Sparse Probing, June 2023.

Adam Karvonen, Can Rager, Samuel Marks, and Neel Nanda. Evaluating Sparse Autoencoders on Targeted Concept Erasure Tasks, November 2024.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models, March 2024. Comment: Code and data at https://github.com/saprmarks/feature-circuits. Demonstration at https://feature-circuits.xyz.

Anish Mudide, Joshua Engels, Eric J. Michaud, Max Tegmark, and Christian Schroeder de Witt. Efficient Dictionary Learning with Switch Sparse Autoencoders, October 2024. Comment: Code available at https://github.com/amudide/switch_sae.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically Interpreting Millions of Features in Large Language Models, December 2024.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving Dictionary Learning with Gated Sparse Autoencoders, April 2024a. Comment: 15 main text pages, 22 appendix pages.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders, August 2024b. Comment: v2: new appendix H comparing kernel functions & bug-fixes to pseudo-code in Appendix J v3: further bug-fix to pseudo-code in Appendix J.