PARSE: AN AUTONOMOUS ALGORITHM FOR MECHANISTIC INTERPRETABILITY BASED ON DYNAMIC, ADAPTIVE SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

The resolution of semantic superposition in Large Language Models (LLMs) currently relies on sparse autoencoders (SAEs) optimized via static L_1 regularization and heuristic warm-up schedules. These static schedules often force concepts into entangled representations and induce semantic hallucination. We propose that representation collapse is not an inherent flaw of SAEs, but an artifact of static latent expansion which lacks quantitative feedback on the condition of the feature covariance matrix. In this paper we present a novel autonomous algorithm that eliminates heuristic warm-up schedules by allowing sparsity to emerge from the dynamic linear algebraic properties of the covariance matrix under the action of the optimizer. PARSE (Parabolic Adaptive Relaxation for Sparse Encoding) continuously monitors the geometric strain of the latent manifold via a matrix-free Krylov approximation of the condition number. When local semantic cross-talk exceeds the LLM's inherent noise floor, the algorithm dynamically spawns orthogonal vectors strictly aligned with the principal eigenvectors of the local residual error matrix. By temporarily substituting the L_1 penalty with a soft orthogonalization penalty, PARSE absorbs dense concepts and safely recycles dead latents without inducing representation shock. This localized expansion reduces the computational profile from quadratic to linear scaling, eliminating GPU bottlenecks and reducing overhead by over 90%. We validate PARSE on the 4096D residual stream of Llama-3(8B) across 10^7 tokens and demonstrate that complex semantic superposition can be isolated with low L2 error under the constraint of strict numerical orthogonality. We further demonstrate that feature death and representation collapse are distinct phenomena. PARSE thus establishes a scalable, autonomous foundation for mechanistic interpretability and the translation of emergent frontier model concepts.

1 INTRODUCTION

The extraction of monosemantic features relies on sparse autoencoders to project a dense residual stream into a high-dimensional latent space, untangling semantic superposition. Current methodologies enforce this sparsity via static L_1 regularization schemes (Bricken et al., 2023). However, recent evaluations by the Google DeepMind mechanistic interpretability team demonstrated that global L_1 scalar warm-up schedules frequently result in dead features or catastrophic representation collapse when scaling to complex downstream tasks. This has caused researchers to deprioritize fundamental SAE architectures (Smith et al., 2025). Furthermore, standard architectures rely on arbitrary, uniform expansion factors (e.g., 2X, 4X, 8X) that waste compute power on empty semantic space while severely under-resolving dense clusters of concepts (Elhage et al., 2022; Templeton et al., 2024).

We propose that these problems are not an inherent flaw of SAEs but rather an artifact of treating latent expansion with a static heuristic protocol that is insensitive to the evolving dynamics of the feature covariance matrix. The PARSE algorithm replaces arbitrary L_1 heuristics with rigorous criteria grounded in linear algebra. Rather than forcing absolute orthogonality and inducing matrix stiffness, PARSE continuously monitors the geometric strain of the latent manifold via the matrix-free Krylov approximation of the condition number (κ). When local semantic cross-talk exceeds the LLM's inherent baseline noise floor, PARSE dynamically spawns new orthogonal vectors specif-

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.

ically tailored to the low-dimensional error subspace, utilizing a geometric progression to prevent representation shock.

Recent advancements have sought to mitigate the representation collapse and feature shrinkage inherent in standard L_1 regularized SAEs. Architectures such as TopK SAEs and JumpReLU SAEs successfully address activation shrinkage by replacing continuous L_1 penalties with discontinuous thresholding or hard-K masking (Gao et al., 2024; Rajamanoharan et al., 2024). However, while these methods improve reconstruction fidelity, they fundamentally remain heuristic static topology models. They operate under a fixed latent budget defined prior to training, forcing the manifold to discard low-frequency features to accommodate high-density concepts. PARSE diverges from these paradigms by treating the latent capacity as a dynamic, continuous medium allowing the topology to physically expand in response to geometric strain rather than merely re-weighting a fixed, tangled basis.

PARSE has its origins in the generation of orthogonal curvilinear grids for the solution of problems in computational fluid dynamics (CFD) (Anonymous, 2026). The grids typically are the solution of elliptic PDEs that map from a computational domain to the physical domain. These mappings are controlled by the metric tensor matrix for the problem (Thompson et al., 1985). When the physical domain contains fixed internal boundaries the mappings are no longer 1:1 and the metric tensor becomes ill-conditioned. To overcome this problem, the elliptic mapping is replaced by a parabolic mapping in which each grid point is assigned an inertia factor based on local orthogonality metrics and discretization errors. In the context of an SAE, the loss landscape curvature is dictated by the metric tensor, $\Sigma = W^T W$ (Amari, 1998). The application of this theory to SAE feature extraction has been described elsewhere (Anonymous, 2026). It should also be noted that these parabolic mappings have been extended to dynamic front-tracking problems in which dynamic local grid refinement is controlled by local numerical errors. This particular aspect of the algorithm is used in PARSE to generate self-consistent orthogonal sub-space expansions that dynamically resolve the local error covariance matrix.

In Section 2 we present the algorithmic foundations for PARSE, followed by a brief discussion of two Toy Model results in Section 3. Section 4 outlines the empirical validation of PARSE on the residual stream of Llama3-(8B) across a streaming corpus of 10^7 tokens. Section 5 presents the major findings of the PARSE algorithm and its contribution to resolving the interpretability crisis.

2 THE PARSE ALGORITHM

To resolve the competing pathologies of representation collapse and capacity starvation, PARSE modifies the standard SAE optimization objective function. We transition from a static, globally uniform penalty to a dynamic, locally adaptive, geometrically constrained framework:

$$\mathcal{L}_{PARSE} = \mathcal{L}_{recon}(\mathbf{X}) + \vec{\Lambda} \cdot \mathcal{L}_{sparsity}(\mathbf{z}) + \gamma \cdot \mathcal{L}_{ortho}(\mathbf{W}) \tag{1}$$

$\mathcal{L}_{recon}$ is the standard L_2 reconstruction error. But in PARSE it is not just used as a loss metric. Rather it is the localized manifold strain tensor that autonomously triggers both local and global subspace expansions. The global scalar penalty function, λ_{L1}, is replaced with a vector penalty function $\vec{\Lambda}$ that is dynamically tightened locally in dense semantic regions according to the loss of diagonal dominance in the covariance matrix. PARSE also introduces an orthogonal penalty function, $\mathcal{L}_{ortho}$. This penalty function may be viewed as a geometric pushback to prevent semantic feature superposition (Bansal et al., 2018).

It is apparent then that equation 1 represents a dynamic training loop that attempts to achieve an equilibrium between the gradient push towards high density data clusters and the orthogonal penalty function designed to prevent clustering. The condition number of the feature covariance matrix is the key metric to monitor. It will plateau at the natural intrinsic conditioning of the data stream.

Thus equation 1 suggests that L_2 loss should not be the only stopping criteria. Otherwise the easiest path for the autoencoder is to simply learn the identity matrix. If feature superposition is to be avoided then orthogonal vectors must be preserved for the steep semantic gradients. PARSE is not a lossless text compressor but rather a latent semantic feature extractor.

2.1 THE TRAINING CYCLE

The training regimen cycles between two distinct states governed by rigorous criteria. During Phase B, the network optimizes for disentanglement using the standard L_1 penalty augmented by $\vec{\Lambda}$:

$$\mathcal{L}_B = \frac{1}{N}\sum_{i=1}^{N} ||x_i - \hat{x}_i||_2^2 + \vec{\Lambda}||z||_1 \tag{2}$$

When the dynamic subspace expansion criteria trigger, the system enters Phase A. In Phase A, the L_1 penalty is temporarily suspended to prevent numerical stiffness. Instead, a soft-orthogonalization penalty is applied to allow the newly introduced vectors to smoothly align with semantic attractors without inducing geometric shock:

$$\mathcal{L}_A = \mathcal{L}_{recon} + \gamma \left(\underbrace{||\Sigma_{new} - I_k||_F^2}_{\text{Internal Orthogonality}} + \underbrace{||\Sigma_{cross}||_F^2}_{\text{Basis Separation}} \right) \tag{3}$$

The algorithm exits Phase A and transitions back to Phase B when either the relative residual reduction plateaus or is reduced by δ_{target} according to equation 4:

$$\frac{L_{2,spawn} - L_2^{(t)}}{L_{2,spawn}} \geq \delta_{target} \vee \left| \frac{L_2^{(t)} - L_2^{(t-1)}}{L_2^{(t-1)}} \right| \leq \epsilon_{tol} \tag{4}$$

where t is an iteration number.

2.2 TRIGGERING SUBSPACE EXPANSION

A subspace expansion is triggered according to the following criteria.

1. PARSE continuously monitors the condition number of the feature covariance matrix: $\kappa(\Sigma)$. To avoid $\mathcal{O}(d^2)$ computational overhead, Krylov subspace methods plus Lanczos iterations are used to extract the extremal eigenvalues using only matrix-vector products (Saad & Schultz, 1986). The condition number is smoothed against stochastic batch variance using an exponential moving average $\overline{\kappa}(t)$. A subspace expansion is triggered when $\frac{\kappa_\infty(t) - \overline{\kappa}(t)}{\overline{\kappa}(t)} \geq \theta_{dynamic}$, where $\theta_{dynamic} = \max(0.10, 3\sigma_\kappa)$ adapts to the background batch noise. An escalating condition number indicates that basis vectors are becoming collinear and that Σ is becoming ill-conditioned. Hence the need for subspace expansion.
2. The other expansion trigger used in PARSE relates to the reconstruction error, $\mathcal{L}_{recon}$ in equation(1). It was often found from experiments that $\kappa(\Sigma)$ remained well-behaved indicating the semantic basis vectors were not collinear. Nevertheless, the reconstruction error continued to grow beyond the initial value. This behavior suggested capacity starvation rather than representation collapse. Under these conditions a geometric strain trigger for expansion was defined as:

   $$L_2(t) \geq (1 + \eta_{strain}) L_{2,max} \tag{5}$$

   where η_{strain} is the critical fraction of the variance ceiling that the reconstruction error must exceed to trigger an expansion.
3. To guard against non-convex optimization jumps into adjacent alien minima (Dauphin et al., 2014), PARSE monitors the trace of the covariance matrix to ensure matrix rank is conserved: $Tr(\Sigma) = ||W_{raw}||_F^2$. Since PARSE enforces continuous column normalization on the decoder matrix such that, $||w_{D,j}||_2 = 1$, the trace of the normalized covariance matrix is constrained to d_h. Hence the matrix-free Krylov approximation used both for the trace and κ tracking is applied to the pre-normalized weight matrix, W_{raw}.

2.3 Methodology for Subspace Expansion

In overcomplete neural representations, the residual reconstruction error, $e = x - \hat{x}$, resides on a compact, low-dimensional error subspace: $\mathcal{S}_e = \text{span}\{e_1, e_2, \ldots, e_m\} \subset \mathbb{R}^{d_{in}}$, where $m \ll d_{in}$ (Elhage et al., 2022). Hence, resolving this error does not require global re-orthogonalization, but rather adaptive local refinement strictly within $\mathcal{S}_e$. Upon triggering an expansion, k new dimensions are appended to the existing, well-conditioned dictionary $W_{old} \in \mathbb{R}^{d_{in} \times d_{old}}$.The expanded decoder is partitioned as $W_D = [W_{old} \quad W_{new}]$. Assuming the sub-block Σ_{old} remains stable, the full Gram matrix is blocked, and the soft-orthogonalization penalty during Phase A is restricted solely to local interactions as seen in equation 3. This equation prevents mutual redundancy among the newly spawned vectors while forcing them to align with the unrepresented error directions in $\mathcal{S}_e$. This formulation thus reduces the computational profile from quadratic growth, $\mathcal{O}((d_{old} + k)^2)$, to linear scaling with respect to the existing basis, $\mathcal{O}(k^2 + k \cdot d_{old})$. For large-scale dictionaries (e.g., $d_{old} = 16,384$), this eliminates GPU memory bottlenecks while preserving absolute numerical stability during adaptive refinement.

As demonstrated in an earlier publication (Anonymous, 2026) (and Toy Model 2), the arbitrary injection of latent vectors into an active space can induce representation shock. Randomly spawning new, uncalibrated dimensions warps the loss landscape, causing discontinuities in the gradient and matrix entanglement. As shown in equation 3, newly spawned vectors are introduced orthogonally. But they must also be properly scaled to absorb new concepts smoothly. By distributing newly spawned vectors strictly along the principal eigenvectors of the local error covariance matrix using a diminishing geometric ratio ($\rho \in (0, 1)$), the new dimensions fan out smoothly. This bounds the perturbation of the metric tensor, cleanly resolving fine semantic features without triggering representation collapse.

When global latent space expansion is required, PARSE addresses the capacity deficit through a two-step process.

1. **Dead Latent Rehabilitation:** The system first identifies existing dead latents that have collapsed into the null space. Instead of abandoning this capacity, PARSE strips these inactive features of their historical weights. They are then dynamically re-injected back into the active space, strictly aligned with the principal eigenvectors of the local error covariance matrix (Σ_{error}) and scaled via a geometric progression to prevent representation shock. This scaling is applied strictly to the newly spawned rows of the encoder matrix (W_E) and its associated biases. This prevents representation shock by dampening initial activation magnitudes systematically. The corresponding newly spawned columns in the decoder matrix (W_D) remain strictly bounded by the continuous unit-norm constraint.
2. **Orthogonal Spawning:** If the reservoir of dead latents is completely exhausted and the error manifold remains unresolved, the system resorts to true capacity expansion. PARSE spawns a brand new set of k orthogonal vectors, augmenting the total hidden dimensionality ($d_h \to d_h + k$) up to a maximum hardware-defined capacity (D_{max}).

   This targeted injection ensures that both recycled and newly spawned dimensions fan out smoothly to map the unrepresented signal without warping the previously established feature covariance matrix.

2.4 Stopping Criteria and Numerical Orthogonality

Unlike Toy Models where the true features are explicitly known, evaluating the success of semantic extraction on a frontier model requires an intrinsic benchmark. We define global convergence as the state at which semantic superposition is considered fully resolved. This mechanism relies on comparing the unresolved variance in the latent feature space against the inherent statistical uncertainty of the foundational language model itself:

- **The Baseline Noise Floor (E_{base}):** In a real-world deployment, further semantic resolution becomes impossible once the residual error drops below the LLM's intrinsic uncertainty limit. We calculate this true noise floor by measuring the cross-entropy loss (next-word prediction) of the uncorrupted Llama-3 model. This cross-entropy baseline establishes our hard target, E_{base}.

- **The Reconstruction Energy (E_{skew}):** Concurrently, we monitor the absolute measure of the unrepresented signal magnitude within the PARSE architecture via the Frobenius norm of the residual error matrix, defined as $||E_{skew}||_F$.

During execution the extraction is deemed globally successful, and the solver permanently halts, strictly when the following condition is satisfied: $||E_{skew}||_F < E_{base}$. Upon crossing this threshold, the PARSE architecture has achieved *Numerical Orthogonality*. The system stabilizes without requiring absolute orthogonalization, and any remaining residual signal is mathematically indistinguishable from the baseline noise of the language model. The pursuit of perfect L_1 sparsity (absolute orthogonality) in the presence of inherent noise leads to over-regularization that destroys genuine semantic meaning. It is counter-productive to force an SAE beyond the LLM's baseline noise floor. By accepting some degree of non-zero off-diagonal elements in Σ a rigorous definition of resolved semantic features is achieved.

2.5 THE PARSE EXECUTION PIPELINE

The PARSE architecture operates as a continuous, self-regulating control loop governed by the topological strain of the latent manifold. During standard optimization, the network continuously evaluates the global L_2 reconstruction error against the dynamic strain threshold (η_{strain}). If the error remains below this boundary, standard training proceeds until ($||E_{skew}||_F < E_{base}$) is satisfied, signifying global convergence and terminating the solver. Conversely, if the strain threshold is breached, the algorithm temporarily halts standard training and engages the capacity management protocol. The system first triggers the recycling of dead latents, scanning the batch for strictly $||z||_1 = 0$; if dormant vectors are identified, they are geometrically resampled directly into the principal eigenvectors of the unresolved error void. If the dormant pool is exhausted and the strain persists, the system defaults to dynamic VRAM expansion, spawning a new geometric progression of orthogonal dimensions. Both capacity interventions immediately route into Phase A—a localized orthogonal relaxation loop that stabilizes the covariance matrix before returning the network to the standard training loop.

2.6 COMPUTATIONAL COMPLEXITY AND ASYMPTOTIC SCALING

A primary barrier to scaling dynamic neural topologies is the quadratic memory and computational cost of global soft-orthogonalization. By partitioning the dictionary into two blocks, (W_{old}) and a dynamic local patch (W_{new}), the PARSE architecture alters the asymptotic scaling profile of the expansion phase.

Evaluating the full Gram matrix penalty across the entire feature space yields a global Frobenius complexity that scales quadratically with the total dictionary size:

$$\text{Global Penalty Complexity: } \mathcal{O}((d_{old} + k)^2) \tag{6}$$

By contrast, the localized subspace orthogonalization restricts the penalty strictly to internal redundancy and cross-basis separation. This localized approach transitions the computational profile to linear scaling with respect to the existing basis:

$$\text{Localized Penalty Complexity: } \mathcal{O}(k^2 + k \cdot d_{old}) \tag{7}$$

This decoupling provides a computational advantage. For large-scale dictionaries where $d_{old} \gg k$ (e.g., $d_{old} = 16,384$ and $k = 512$), this localized formulation reduces the required term evaluations by over 90%. Consequently, PARSE reduces the GPU memory bottlenecks that typically restrict dynamic autoencoders.

2.7 OPERATIONAL HYPERPARAMETERS AND SYSTEM SENSITIVITY

Unlike static L_1 schedules that rely on heuristic scalar warm-ups, the PARSE architecture is governed by a set of explicit, geometrically derived boundary conditions. These user-specified hyperparameters control the sensitivity of the subspace expansion triggers, the scale of the topological expansion, and the strictness of the localized orthogonalization.

Table 1 summarizes the core operational parameters utilized during empirical calibration.

Table 1: PARSE Operational Hyperparameters and Boundary Conditions

Parameter	Value	Geometric Function
η_{strain}	0.15	**Geometric Strain Trigger:** The critical fraction of the variance ceiling ($L_{2,max}$) that the reconstruction error must exceed to trigger an expansion.
k_{spawn}	512	**Subspace Expansion Size:** The discrete number of orthogonal basis dimensions dynamically appended during a resolution loop.
$\theta_{dynamic}$	$\max(0.10, 3\sigma_\kappa)$	**Dynamic Sensitivity Criterion:** The required relative rate of change in the condition number ($\Delta\kappa$) to trigger a reset, adapting to background batch noise.
δ_{target}	0.50	**Relative Error Reduction:** The strict threshold for residual reduction required to exit Phase A and return to Phase B sparsity.

3 Synthetic Toy Models

As emphasized in Elhage et al. (2022), Toy Models are important for validating new algorithms. In this spirit, before we test PARSE on a large pre-trained LLM we must prove the model in a fully controlled synthetic environment. We need to construct a model where we artificially induce the exact numerical pathologies that plague real-world neural networks. These include ill-conditioning, matrix stiffness, semantic superposition and representation shock.

The goal of the first Toy Model is to force a small set of orthogonal features to compress into a lower-dimensional non-orthogonal space, and then to perfectly reconstruct the original orthogonal features without causing feature death or representation collapse. The methodology, theory and Pytorch script are presented in an earlier publication (Anonymous, 2026). To summarize, we generated a highly sparse dataset of 100 independent orthogonal features and then projected them into a 20D space. Hence they were forced into non-orthogonal superposition creating an ill-conditioned covariance matrix. Gaussian noise was injected into the data projection to simulate baseline cross-entropy loss. As shown in (Anonymous, 2026), the PARSE algorithm successfully minimized the L_2 reconstruction loss and untangled the covariance matrix recovering the true decoder basic vectors. This Toy Model demonstrated two important features of PARSE for mechanistic interpretability that were implemented in the full algorithm. The first is the importance of a soft orthogonalization (Phase A) prior to implementing the sparsity penalty (Phase B) so as to avoid early feature death. The second lesson is the recognition that perfect orthogonality is counter-productive when it forces an SAE beyond the LLM's baseline noise floor.

Toy Model 2 is designed to study representation shock caused by the sudden injection of new latent dimensions in response to a dense tangled alien concept in the data stream (Chen et al., 2016). The model details and code are also presented in (Anonymous, 2026). Two different scenarios are presented. In scenario A, five new latent vectors are spawned using standard uncalibrated random initialization (Glorot & Bengio, 2010). In scenario B, PARSE spawns five new latent vectors using a geometric orthogonal progression that fans out smoothly along the isolated error vector. The results indicated that scenario A ended with representation collapse caused by an initial discontinuity in the $\|\nabla\mathcal{L}\|_2$ gradient magnitude. The gradient magnitude in scenario B however remained low and bounded. This demonstrates that the PARSE expansion created a smooth, continuous bridge between the existing latent dimensions and the newly spawned capacity. This Toy Model mirrors the pathologies seen in modern static SAEs that attempt to force uncalibrated dead latents back into an active space to untangle dense residual streams (Smith et al., 2025; Bricken et al., 2023).

4 Empirical Validation: Large-Scale Semantic Extraction on Llama-3

To evaluate the stability, capacity, adaptation and semantic fidelity of the PARSE algorithm against standard static baseline models, we conducted calibrated benchmarks on the residual stream of

Llama-3(8B) (Layer 16) across a streaming Wikipedia corpus of 10^7 tokens (AI@Meta, 2024; Nanda & Bloom, 2022). Recent papers in the interpretability community, such as the SAEBench framework, have demonstrated that SAE training methods stabilize and become statistically apparent with the first 10 to 25 million tokens (Karvonen et al., 2024). All runs were performed on the RunPod A100-SXM 80GB VRAM GPU.

It is well known that dynamically altering the dictionary dimension mid-training or initializing new latents in an active streaming pipeline is prone to representation collapse (Chen et al., 2016). In addition to this metric, our benchmarks also measure intrinsic geometric stability of the covariance matrix (condition number), the L_0 activation density, dead latent capacity and prompt response fidelity (hallucinations). The Pytorch code is available in Appendix A.

Table 2 summarizes the results of this study.

Table 2: Model Comparisons across 10^7 Tokens on Llama-3 (Layer 16)

Model	**Final L_2 Error**	**Condition No. (κ)**	**L_0 Density**	**Dead Latents**
Static Baseline (1536D)	0.0708	203.56	132.3	371
PARSE Dynamic (1536D)	0.2251	68.65	113.1	381
Static Baseline (3072D)	0.0552	954.60	121.7	1388
PARSE Dynamic (3072D)	0.2088	265.21	131.0	1031

The "Static Baseline (1536D)" model was initialized at 1536D and ran with no dynamic expansions. The "PARSE Dynamic (1536D)" model was initialized at 512D and underwent several subspace expansions. The "Static Baseline (3072D)" model was initialized at 3072D and again ran with no dynamic expansions or dead latents recycling. Finally, the PARSE Dynamic (3072D) was initialized at 2560D and expanded once though it did recycle dead latents throughout the run (as did PARSE Dynamic (1536D)). We next discuss the key results of these models.

4.1 Importance of not optimizing on L_2 error alone

A key finding of these runs is that while the static baseline models consistently achieve lower L_2 reconstruction errors, they do so at the expense of polysemanticity, sacrificing true feature separation. This is evidenced by their condition numbers and hallucinations to the prompt. By not allowing for dynamic subspace expansion and orthogonalization, the spaces become warped. In contrast, PARSE optimizes for geometric disentanglement by enforcing active sparsity and local orthogonalization. While PARSE did not complete the prompts verbatim, its responses did align with the semantic meaning of the prompts. These results emphasize the importance of not optimizing for low L_2 errors exclusively.

4.2 Feature death and representation collapse are distinct phenomena

While the current mechanistic interpretability literature conflates dead latents (inactive features) with a failed optimization state or representation collapse (Smith et al., 2025), our model results suggest they are distinct phenomena governed by different geometric mechanisms. Both the static and PARSE 1536D model runs exhibited near-identical rates of inactive capacity (371 and 381 dead latents respectively). However the geometric states of the surviving active dimensions diverged. The static model's high condition number (203) reflects a loss of orthogonality among the surviving active features which resulted in semantic hallucination. PARSE however shed the same number of excess dimensions while maintaining an orthogonal, well-conditioned active feature space ($\kappa = 68$) that preserved the domain-specific logic. By treating the minimization of dead latents as a primary evaluation metric, current static L_1 schedules may force the optimizer to warp the feature covariance matrix to keep latents active inducing representation collapse.

4.3 Semantic fidelity vs rote memorization

As shown in Table 2, the PARSE Dynamic (3072D) model with dead latent resampling ended with 1031 dead latents and a condition number of 265. But when we ran the same model with no dead latent resampling the number rose to 1788 and the condition number nearly doubled to 510. These

numbers would suggest polysemantic clusters of concepts. Yet when we ran the test prompt on this model it correctly predicted the next word while the model with resampling did not. Instead the PARSE Dynamic model with resampling responded to the prompt in the same "spirit" of the meaning but framed it in its own words. Further, when the prompt was provided multiple times the verbatim response was always different but retained the same contextual meaning while the first model always returned the same word. Both models ended with similar values for L_0 and L_2. A close examination of the cosine similarity for both models revealed that the first model without dead latent resampling had large numbers of nearly collinear features while the model with resampling preserved near-perfect orthogonality. These results suggest that the first model with 58% dead latents is just copying the input rather than discovering the underlying concepts. Because the 42% active vectors are not independent, they activate together in a correlated block. The model simply memorized the Wikipedia text but failed to understand the concepts. The dynamic PARSE model with resampling however built a dictionary of basic ideas with its surviving orthogonal latents (67%) and decomposed the residual stream into true, independent semantic building blocks.

4.4 Activation sweep of PARSE Dynamic (3072D) model

To further understand the internal state of the this model, which included dead latent recycling, we performed an activation sweep to see how individual dimensions aligned with specific concepts. To achieve this we passed a few thousand tokens through the network and recorded which specific words caused which specific dimensions to spike. The results reflected a well-calibrated sparse autoencoder. For example, dimension 0558 activated solely on numbers while dimensions 0323 and 1723 activated on the structural grammar of a new sequence initiation, i.e. when a fresh thought or context window is beginning. We next confirmed that the dynamically spawned vectors in PARSE actually achieve mutual orthogonality by intercepting the training loop and computing pairwise cosine similarities across localized basis vectors. The training loop was intercepted following the Phase A soft-orthogonalization relaxation window. The mean off-diagonal numbers for this model were 0.012 and the 0.110 for the maximum absolute correlation. Corresponding numbers for the model with no dead latent recycling were 0.992 and 0.997 indicating significant feature superposition. These data confirms that by maintaining a near-zero off-diagonal cosine similarity, the newly injected dimensions successfully fanned out to resolve the local error covariance matrix (Σ_{error}) without collapsing into redundant clusters. This structural independence is the mechanistic driver that allows for the untangling of complex semantic domains without hallucination. We will present further information on this problem in a future paper.

4.5 L1 shrinkage and the PARSE solution

L_1 shrinkage is a well-known side effect of using the L_1 norm to enforce sparsity (Bricken et al., 2023). In standard SAEs, the global loss function in equation 1 does not contain the orthogonalization term. Since the goal of the L_1 penalty function is to push the network to use as few latent features (z) as possible, the optimizer applies downward pressure on every active feature regardless of its importance to the reconstruction. This causes the network to shrink the true magnitude of the activation which results in several problems:

- The reconstructed output is smaller in magnitude than the input so the model never recovers the original signal.
- Weakly activated but valid features are deactivated causing feature death.
- To compensate for the shrinking activations, the decoder weights are often scaled up to boost the signal which can destabilize the covariance matrix.

Current attempts to mitigate this problem, such as TopK and JumpReLU SAEs have had success but they remain heuristic static models (Gao et al., 2024; Rajamanoharan et al., 2024). They operate under a fixed latent budget defined prior to training. PARSE on the other hand treats the latent capacity as a dynamic space that can physically expand in response to geometric strain rather than re-weighting a fixed tangled basis. Most importantly, as shown in Section 2, the addition of the orthogonality penalty function in equation 1 counteracts the L_1 penalty function from driving latent death and superposition. Instead it attempts to achieve an equilibrium between the gradient pull towards sparsity and high density data clusters and the orthogonality push designed to prevent clustering and representation collapse.

While PARSE ameliorates the L_1 shrinkage problem by forcing a natural equilibrium between sparsity and orthogonality, it also reduces the shrinkage problem by reinjecting dead latents in a rigorous targeted adaptive process. However, numerical experiments to be reported later, are demonstrating that zero dead latents will result in other pathological problems. Early experiments that use the Huber function to smooth the L_1 penalty and prevent all latent death trigger the "Zombie Vector Trap". Since vectors never die the space fills with large numbers of small, non-zero "zombie" activations that destroy strict local sparsity (L_0) replacing specialized feature extraction with a dense blurry neural smear.

5 CONCLUSIONS

The PARSE algorithm was developed to address semantic superposition in LLMs that rely on SAEs based on static L_1 regularization and heuristic warm-up schedules. The lack of a rigorous mathematical theory for latent space expansion and the lack of a rigorous stopping criteria to assess the quality of the optimizer results have contributed to the interpretability crisis. Linear algebraic principles that were used to treat ill-conditioned metric tensors in CFD mapping problems have proven to be relevant to identifying and addressing the current problem in SAE implementation. We have demonstrated in this paper that representation collapse is not an inherent flaw of SAEs but rather an artifact of static algorithms and the lack of quantitative metrics to act as feedback on the quality of feature representation. At the heart of PARSE is equation 1 which proposes that the global objective loss function must include a counter-balance to the sparsity penalty in the form of an orthogonality penalty function. In addition, equation (5) provides a geometric strain trigger for feature space expansion, which follows a strict targeted algorithm to prevent superposition during token streaming. The main results of this study are as follows.

1. Optimization on the L_2 error alone may lead to polysemanticity and hallucinations unless the subspace is allowed to dynamically expand.

2. Feature death and representation collapse are distinct pathologies. Current static L_1 schedules conflate them because their primary metric is low L_2 error and low dead latents. Representation collapse has been seen to result from the focus on this metric alone.

3. Current L_2 global minimization procedures inherently prioritize dominant features (Bricken et al., 2023). This often results in a high percentage of dead latents. Without calibrated dead latent resampling, the active latents may become collinear and simply memorize the streaming text rather than building a dictionary of basic ideas.

4. L_1 shrinkage is a consequence of static L_1 sparsity schedules. By including an orthogonality penalty function in the global optimizer, feature death and semantic superposition may be reduced.

5. By performing targeted localized subspace expansion using only matrix-vector products, PARSE reduces the computational profile from quadratic to linear scaling. This makes it suitable for large frontier models.

In summary, PARSE has identified one of the causes of semantic superposition in LLMs that rely on static SAEs. A methodology based on linear algebra has been presented to address this problem in current SAEs. The result is a conceptual model that guards against representation collapse and semantic hallucination - primary objectives of mechanistic interpretability.

NOMENCLATURE

The following symbols and variables are utilized throughout the formulation of the PARSE model:

Symbol	Definition
Network Dimensions & Data	
$x, \hat{x}$	Input residual stream vector and reconstructed output vector
d_{in}	Input dimensionality of the dense residual stream
d_h	Hidden dimensionality of the latent feature space (dynamic)
$\mathbf{z}$	Latent activation vector
W_E, W_D	Encoder projection matrix and Decoder metric tensor
PARSE Objective & Loss Functions	
$\mathcal{L}_{PARSE}$	Total dynamic optimization objective
$\mathcal{L}_{recon}$	Global L_2 reconstruction error
$\vec{\Lambda}$	Vectorial sparsity penalty (dimension-specific L_1 pressure constraint)
γ	Scaling coefficient for the localized soft-orthogonalization penalty
$\mathcal{L}_{ortho}$	Localized soft-orthogonalization constraint applied during expansion
Dynamic Geometry & Hyperparameters	
Σ_{error}	Local error covariance matrix of the unresolved residual
κ	Matrix condition number via Krylov-Lanczos approximation
η_{strain}	Relative L_2 strain threshold triggering localized subspace expansion
δ_{target}	Target relative error reduction for Mini-Phase A early stopping
Localized Expansion & Resampling	
v_{err}	Principal eigenvectors extracted from Σ_{error}
ρ	Diminishing geometric progression ratio ($\rho \in (0, 1)$)
c	Initial geometric scaling constant for spawned or resampled latents

REFERENCES

AI@Meta. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Anonymous. Parse: Parabolic adaptive relaxation for sparse encoding. *ResearchGate*, 2026.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Trenton Bricken, Adamarie Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Anthropic*, 2023.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *Anthropic*, 2022.

Leo Gao et al. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Adam Karvonen et al. Saebench: A framework for evaluating sparse autoencoders. *arXiv*, 2024.

Neel Nanda and Joseph Bloom. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`, 2022.

Senthooran Rajamanoharan et al. Improving dictionary learning with jumprelu. *arXiv preprint arXiv:2407.14435*, 2024.

Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

Logan Smith, Senthooran Rajamanoharan, Arthur Conmy, Callum McDougall, Jan Kramar, Tom Lieberum, Rohin Shah, and Neel Nanda. Negative results for saes on downstream tasks and deprioritising sae research. LessWrong, 2025. URL `https://www.lesswrong.com/posts/4uXCAJNuPKtKBsi28/negative-results-for-saes-on-downstream-tasks`.

Adamarie Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Leblond, Andy Bailie, Saurav Sharma, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Anthropic*, 2024.

Joe F Thompson, Zahir UA Warsi, and C Wayne Mastin. *Numerical Grid Generation: Foundations and Applications*. North-Holland, 1985.

A Appendix A: PyTorch Implementation of the PARSE Algorithm

The following script details the core PARSE architecture, including the dynamic Gram-Schmidt orthogonal expansion and the localized condition number approximation utilized for the Llama-3(8B) empirical validation.

```python
import torch
import torch.nn as nn
import torch.optim as optim

class LocalizedPARSE(nn.Module):
    def __init__(self, d_in, d_initial=512):
        super().__init__()
        self.d_in = d_in
        self.d_h = d_initial
        self.W_E = nn.Parameter(torch.randn(self.d_h, d_in) * 0.02)
        self.b_E = nn.Parameter(torch.zeros(self.d_h))
        self.W_D = nn.Parameter(torch.randn(d_in, self.d_h) * 0.02)
        self.normalize_decoder()

    def normalize_decoder(self):
        with torch.no_grad():
            self.W_D.data = self.W_D.data / torch.norm(self.W_D.data, dim=0, keepdim=True).
                clamp(min=1e-8)

    def forward(self, x):
        z = torch.relu(x @ self.W_E.T + self.b_E)
        x_hat = z @ self.W_D.T
        return x_hat, z

    def spawn_dimensions(self, k=512):
        new_WE = torch.randn(k, self.d_in, device=self.W_E.device) * 0.02
        new_bE = torch.zeros(k, device=self.b_E.device)
        new_WD = torch.randn(self.d_in, k, device=self.W_D.device) * 0.02

        with torch.no_grad():
            new_WD = new_WD / torch.norm(new_WD, dim=0, keepdim=True).clamp(min=1e-8)
            self.W_E = nn.Parameter(torch.cat([self.W_E.data, new_WE], dim=0))
            self.b_E = nn.Parameter(torch.cat([self.b_E.data, new_bE], dim=0))
            self.W_D = nn.Parameter(torch.cat([self.W_D.data, new_WD], dim=1))
            self.d_h += k

def estimate_condition_number_lanczos(W, k_iter=15):
    with torch.no_grad():
        d_h = W.shape[1]
        v = torch.randn(d_h, device=W.device, dtype=W.dtype)
        v = v / (torch.norm(v).clamp(min=1e-8))
```

```
        alpha = torch.zeros(k_iter, device=W.device, dtype=W.dtype)
        beta = torch.zeros(k_iter, device=W.device, dtype=W.dtype)
        v_prev = torch.zeros_like(v)

        for j in range(k_iter):
            w = torch.matmul(W.t(), torch.matmul(W, v))
            alpha[j] = torch.dot(v, w)
            w = w - alpha[j] * v - (beta[j-1] * v_prev if j > 0 else 0)

            beta_j = torch.norm(w)
            if beta_j < 1e-6: break

            if j < k_iter - 1:
                beta[j] = beta_j
                v_prev = v
                v = w / beta_j

        T = torch.diag(alpha[:j+1]) + torch.diag(beta[:j], 1) + torch.diag(beta[:j], -1)
        eigvals = torch.linalg.eigvalsh(T)

        if len(eigvals) > 0 and eigvals[0] > 1e-6:
            return (eigvals[-1] / eigvals[0]).item()
        return float('inf')

def parse_training_loop(parse_net, optimizer, fetch_batch_fn):
    tokens_processed = 0
    sensor_arm_token = 500_000
    L2_baseline = float('inf')
    eta_strain = 0.15

    while tokens_processed < 10_000_000:
        parse_net.train()
        X_norm = fetch_batch_fn()
        optimizer.zero_grad()

        x_hat, z = parse_net(X_norm)
        L2_loss = torch.mean(torch.sum((X_norm - x_hat)**2, dim=-1))
        L_sparse_loss = 1e-3 * torch.mean(torch.sum(torch.abs(z), dim=-1))

        loss = L2_loss + L_sparse_loss
        loss.backward()
        optimizer.step()

        with torch.no_grad():
            parse_net.W_D.data = parse_net.W_D.data / (torch.norm(parse_net.W_D.data, dim=0,
                 keepdim=True).clamp(min=1e-8))

        current_L2 = L2_loss.item()
        tokens_processed += X_norm.shape[0]

        if tokens_processed < sensor_arm_token:
            L2_baseline = current_L2
        else:
            L2_baseline = min(L2_baseline, current_L2)

        # Trigger Evaluation & Subspace Expansion
        if tokens_processed >= sensor_arm_token and current_L2 > (1.0 + eta_strain) *
             L2_baseline:
            with torch.no_grad():
                dead_mask = (torch.sum(torch.abs(z), dim=0) < 1e-6)
                dead_indices = torch.nonzero(dead_mask, as_tuple=True)[0]

            if len(dead_indices) > 0:
                residuals = X_norm - x_hat
                Sigma_error = torch.matmul(residuals.t(), residuals) / X_norm.shape[0]
                eigenvalues, eigenvectors = torch.linalg.eigh(Sigma_error)
                v_err_top = eigenvectors[:, -32:]

                for i, idx in enumerate(dead_indices[:128]):
                    v_target = v_err_top[:, i % 32]
                    r = torch.randn_like(v_target)
                    u_i = r - torch.dot(r, v_target) * v_target
                    u_i = u_i / (torch.norm(u_i) + 1e-8)

                    beta_i = min(0.99, 0.8 * (0.9 ** ((i // 32) % 15)))
                    alpha_i = (1.0 - beta_i**2)**0.5

                    parse_net.W_D.data[:, idx] = alpha_i * v_target + beta_i * u_i

            elif parse_net.d_h < 16384:
```

```
                parse_net.spawn_dimensions(k=512)

            optimizer = optim.Adam(parse_net.parameters(), lr=1e-3)
            sensor_arm_token = tokens_processed + 250_000
            L2_baseline = current_L2
```