

Beyond Dense States: Sparse Transcoders as Causally Testable Operators for LLM Latent Reasoning

Yadong Wang^{1,2}, Haodong Chen^{1,2}, Yu Tian³,
Chuanxing Geng¹, Dong Liang¹, Xiang Chen^{1,2*}

¹ Nanjing University of Aeronautics and Astronautics

²State Key Laboratory of Ocean Sensing, ZJU-Hangzhou Global Scientific and Technological Innovation Center, Zhejiang University, Hangzhou, 311215, China

³360 Security Capability Center

{adamwang373, xiang_chen}@nuaa.edu.cn

Abstract

Latent reasoning reduces the token-generation cost of chain-of-thought reasoning by replacing explicit intermediate tokens with continuous latent transitions. However, existing latent reasoning methods usually rely on dense and entangled transitions, making their reasoning trajectories difficult to inspect or intervene on. We introduce **LSTR** (Latent Sparse Transcoder Reasoning), a framework that turns sparse transcoders from post-hoc diagnostic tools into in-loop, intervenable transition components for latent reasoning. At each latent step, a **Latent Transition Transcoder (LTT)** combines a linear skip path with a Top- k sparse innovation path, exposing a small set of active sparse features. Under matched compression settings, LSTR offers a mechanistically inspectable alternative to dense latent reasoning. On GSM8K-Aug, ablating only a few top-active sparse features reduces accuracy by up to 16.5%, whereas analogous interventions have much smaller effects in dense latent baselines. These results indicate that the active sparse features are causally involved in the latent transition process, rather than merely post-hoc descriptors. Additional experiments on mathematical benchmarks and StrategyQA suggest that sparse latent transitions can preserve the compression benefits of latent reasoning while making the resulting trajectories more inspectable and intervenable.

1 Introduction

Chain-of-Thought (CoT) prompting has substantially improved the reasoning capabilities of LLMs by decomposing complex tasks into explicit intermediate steps (Xiang et al., 2025; Wei et al., 2022; Jaech et al., 2024; Wang et al., 2025). However, this paradigm introduces a significant computational bottleneck. Extended token-level reasoning chains increase inference latency and memory con-

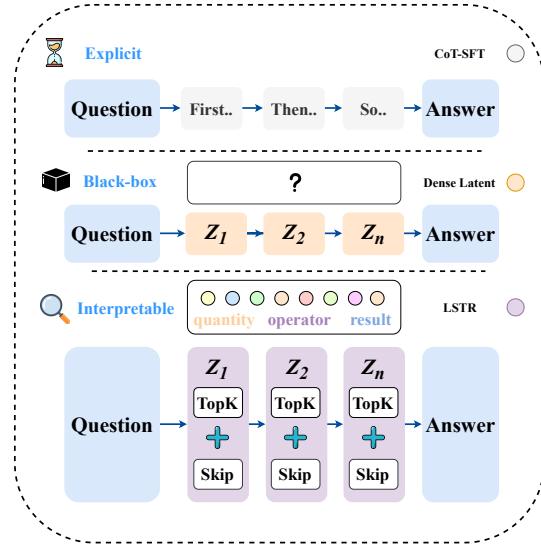


Figure 1: Comparison of reasoning paradigms. Explicit CoT uses long token sequences, while dense latent reasoning compresses intermediate computation into opaque continuous states. LSTR computes each latent transition through a bounded set of Top- k sparse features, exposing transition-level units for inspection and intervention.

sumption, thereby restricting deployment in time-sensitive and resource-constrained environments. Because reasoning depth is tightly coupled with autoregressive token generation, the computational cost scales approximately linearly with the number of reasoning steps, rapidly becoming prohibitive for complex tasks.

To address this limitation, recent studies have explored latent reasoning, in which intermediate reasoning steps are compressed into continuous hidden representations rather than explicit language tokens (Chen et al., 2022; Saunshi et al., 2025; Chen et al., 2025b,a). Methods such as Coconut (Hao et al., 2024) and CoLaR (Tan et al., 2025) demonstrate that multiple reasoning steps can be compressed into a small number of latent transitions,

*Corresponding author.

thereby substantially reducing overall inference costs. Despite these computational efficiency gains, existing latent reasoning methods remain fundamentally difficult to interpret. Most approaches rely on dense latent representations, in which all latent dimensions remain active throughout the entire reasoning process. Consequently, the internal reasoning dynamics become mechanistically opaque. Specifically, there is no principled mechanism for identifying which latent dimensions contribute to specific reasoning operations, a structural entanglement that severely limits the ability to inspect or intervene on latent reasoning trajectories.

In parallel, advances in **mechanistic interpretability** have shown that dense neural representations can often be decomposed into sparse sets of semantically meaningful features. Sparse autoencoders (Bricken et al., 2023) and transcoder models (Dunefsky et al., 2024; Paulo et al., 2025) have uncovered interpretable semantic structures in frozen language models. Nevertheless, these methods remain primarily **post-hoc**. Sparse features are extracted from pretrained activations for diagnostic analysis, but they do not directly participate in the reasoning process during inference.

This separation exposes a fundamental gap between two research directions. Latent reasoning improves computational efficiency but sacrifices interpretability, whereas sparse feature learning improves interpretability while remaining disconnected from reasoning dynamics. This gap motivates a central question: *can compressed latent reasoning be parameterized through a sparse transition interface that remains competitive with dense latent transitions while supporting direct feature-level intervention?*

This work provides an affirmative answer to this question by introducing LSTR (Latent Sparse Transcoder Reasoning), a framework that reformulates latent reasoning as a sequence of transitions through a sparse feature space. In contrast to conventional dense latent reasoning, LSTR imposes an explicit sparsity bottleneck at each latent step, ensuring that only a small subset of features is activated to contribute to each latent transition update. The core component of LSTR is the LTT, which integrates a linear path with a sparse path. This design decouples background manifold transport from sparse residual updates. Consequently, sparsity in LSTR is not merely a regularization strategy applied after the reasoning computation, but a fundamental architectural principle governing how rea-

soning computations are distributed and interpreted across latent dimensions.

A direct consequence of this formulation is **sparsity-mediated capacity control**. The sparsity budget k determines the number of active sparse features during inference, thereby providing a mechanism for regulating transition capacity without retraining. More importantly, the sparse feature structure of LSTR enables direct causal analysis of latent reasoning dynamics. Controlled ablation experiments demonstrate that removing a small number of highly activated sparse features substantially degrades reasoning performance in LSTR, whereas the same intervention produces negligible effects in dense latent baselines. These findings indicate that the active sparse features are not merely diagnostic representations, but causally involved transition components.

Rather than treating latent reasoning only as a compression mechanism, we study whether its transition process can be given a sparse and causally testable interface. In this view, explicit CoT serves as an uncompressed reference, while the central comparison is whether sparse latent transitions can preserve the accuracy–length trade-off of dense latent reasoning while exposing feature-level intervention points. Experiments on mathematical reasoning benchmarks, including GSM8K-Aug, GSM-Hard, SVAMP, MultiArith, and MATH, as well as the commonsense reasoning benchmark StrategyQA, demonstrate that LSTR maintains competitive performance under latent compression. Furthermore, the proposed approach exposes active sparse features that can be directly inspected, ablated, and causally tested. The primary contributions of this work are summarized as follows:

- We introduce **LSTR**, a sparse latent reasoning framework that uses in-loop transcoders as transition operators rather than post-hoc analysis modules.
- We provide quantitative causal evidence that a small set of active sparse features functionally supports latent reasoning, as targeted ablation produces substantially larger degradation than random or dense-state ablation.
- We show that the sparsity budget k provides an inference-time mechanism for regulating latent transition capacity, enabling feature-level inspection and intervention without retraining.

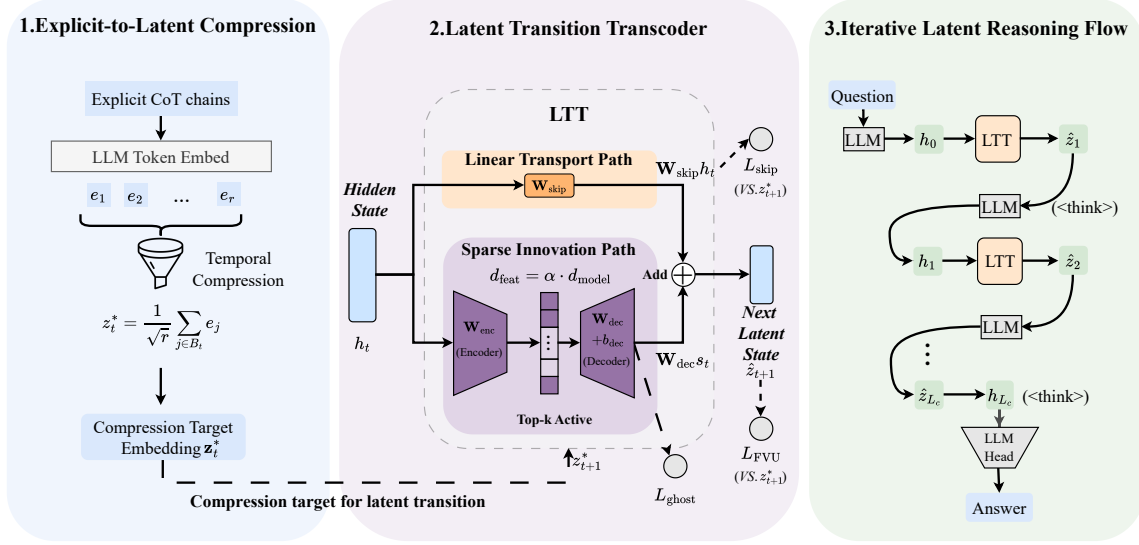


Figure 2: Overview of LSTR. Contiguous explicit reasoning tokens are temporally aggregated into a compressed target embedding. At each latent step, the Latent Transition Transcoder (LTT) bilaterally decomposes transition dynamics: a linear Transport Path maintains manifold continuity, while a Sparse Innovation Path injects bounded sparse updates via Top- k selection. Under a fixed (r, k) budget, LSTR exposes bounded active sparse features per step, facilitating inspection and intervention along the latent trajectory.

2 Related Work

2.1 Latent and Implicit Reasoning

Explicit CoT generation incurs substantial computational overhead and latency (Wei et al., 2022). To address this limitation, **latent reasoning** internalizes the reasoning process into continuous hidden representations (Zhu et al., 2025). While early methods such as iCoT (Deng et al., 2024) absorb reasoning supervision directly into model parameters, recent approaches explicitly model latent trajectories. For example, Coconut (Hao et al., 2024) introduces a “chain of continuous thought” by recursively feeding hidden states back into the model, thereby enabling exploration across latent reasoning paths. Furthermore, CODI (Shen et al., 2025) aligns the latent states of a student model with the reasoning hidden states of a teacher model via self-distillation. CoLaR (Tan et al., 2025) subsequently implements dynamic compression and optimizes these trajectories using reinforcement learning. Despite their effectiveness in reducing inference costs, these approaches rely almost exclusively on **dense** latent representations, wherein all latent dimensions remain jointly active during each transition. This dense formulation inherently limits interpretability and controllability, as individual latent dimensions lack clear semantic attribution. In contrast, the proposed framework introduces an

explicit sparse semantic bottleneck, enabling latent transitions to be composed of localized and interpretable functional features.

2.2 Sparse Transcoders and Feature Learning

Sparse Autoencoders (SAEs) disentangle polysemantic neurons into interpretable semantic features (Bricken et al., 2023). More recently, **Transcoders** extend this paradigm by modeling the functional input–output behavior rather than reconstructing static activations (Dunefsky et al., 2024). Empirical studies show that transcoders capture substantially more interpretable features by directly approximating the operational semantics of neural submodules (Paulo et al., 2025). Architectural refinements such as affine residual paths alleviate the rank limitations imposed by sparse bottlenecks, while ghost-gradient updates revive underutilized features by propagating residual signals to inactive units. However, existing sparse feature learning remains primarily a post-hoc diagnostic tool for static and frozen models. In contrast, our work repurposes sparse transcoders as *active reasoning operators*, with sparse features participating directly in latent state transitions rather than describing them post-hoc. The key distinction is not merely that LSTR produces sparse activations, but that these activations are part of the state transition that constructs the next latent reasoning state.

3 Methodology

3.1 Overview of LSTR

We introduce LSTR (Latent Sparse Transcoder Reasoning), a framework parameterizing latent reasoning as a sequence of sparse transitions. Existing latent reasoning methods replace explicit CoT tokens with continuous latent transitions, but these transitions are typically dense: all dimensions remain active, making individual component contributions difficult to inspect or intervene on. LSTR retains the compressed latent setup but redesigns the transition function. At each latent step, the model computes the subsequent continuous state using a small set of active sparse features.

Given a backbone hidden state, the core LTT module predicts the next latent state via a linear skip path and a sparse innovation path. The skip path facilitates the linear transport of predictable variations between consecutive states, whereas the sparse path models residual nonlinear updates using a bounded set of dictionary features. Rather than assuming perfect separation, this architectural inductive bias establishes a sparse transition interface that facilitates evaluation through feature ablation and intervention. Unlike post-hoc feature extraction, LSTR generates sparse activations internally within the latent reasoning loop to construct the subsequent state directly. Furthermore, the sparsity budget k constrains the number of active features per step, thereby enabling capacity control at inference time without requiring retraining (Fig. 2). LSTR provides an architectural substrate for interpretability by exposing bounded active features; the semantic mapping of these features remains an open empirical question that our framework enables but does not resolve.

3.2 Latent Trajectory Construction and Inference

Given an explicit reasoning sequence T_r of length L_r , LSTR maps it to a compressed latent trajectory $\mathcal{Z}^* = \{\mathbf{z}_1^*, \dots, \mathbf{z}_{L_c}^*\}$ with a compression ratio r , where $L_c = \lceil L_r/r \rceil$. Each latent target is derived by aggregating a contiguous block of token embeddings $\mathcal{B}_t$:

$$\mathbf{z}_t^* = \frac{1}{\sqrt{r}} \sum_{j \in \mathcal{B}_t} \mathbf{e}_j. \quad (1)$$

The $1/\sqrt{r}$ scaling factor stabilizes the variance as r varies, preventing the latent magnitude from scal-

ing with the block size. To ensure numerical stability, the latent targets are further standardized using empirical embedding statistics.

During training, we optimize over the concatenated sequence $[\mathbf{E}_q, \langle \text{think} \rangle, \mathcal{Z}^*, \langle \text{think} \rangle, \mathbf{E}_a]$. At latent step t , the LTT predicts $\hat{\mathbf{z}}_{t+1}$ from the backbone hidden state $\mathbf{h}_t$, and this prediction is supervised by $\mathbf{z}_{t+1}^*$. Concurrently, the language modeling head predicts a randomly sampled token from the corresponding block $\mathcal{B}_{t+1}$. This auxiliary token objective anchors the latent transition to discrete language semantics without requiring complete token-level reasoning generation. Furthermore, random sampling within each block mitigates positional bias.

At inference time, explicit intermediate reasoning tokens are replaced by iterative continuous transitions. Given a question prefix followed by $\langle \text{think} \rangle$, the backbone generates an initial hidden state $\mathbf{h}_0$. The LTT maps $\mathbf{h}_0$ to a sparse activation vector $\mathbf{s}_0$ and a latent state $\hat{\mathbf{z}}_1$. This latent state is then fed back into the backbone as a continuous reasoning input to produce the subsequent hidden state $\mathbf{h}_1$. This procedure is repeated for L_c latent steps. Finally, the model exits the latent reasoning segment and generates the final answer using the language modeling head. Formally, this iterative process is defined as:

$$\begin{aligned} \mathbf{h}_t &= f_\theta(\mathbf{x}_{\leq t}), \\ (\mathbf{s}_t, \hat{\mathbf{z}}_{t+1}) &= \text{LTT}(\mathbf{h}_t), \\ \mathbf{x}_{t+1}^{\text{lat}} &\leftarrow \hat{\mathbf{z}}_{t+1}, \end{aligned} \quad (2)$$

where f_θ denotes the backbone language model and $\mathbf{x}_{t+1}^{\text{lat}}$ represents the continuous latent input consumed at the next latent step.

3.3 Latent Transition Transcoder

The LTT predicts the next latent state $\hat{\mathbf{z}}_{t+1}$ from the backbone hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{\text{model}}}$:

$$\hat{\mathbf{z}}_{t+1} = \mathbf{W}_{\text{skip}} \mathbf{h}_t + \mathbf{W}_{\text{dec}} \mathbf{s}_t + \mathbf{b}_{\text{dec}}, \quad (3)$$

$$\mathbf{s}_t = \text{Top-}k(\sigma(\mathbf{W}_{\text{enc}}(\mathbf{h}_t - \boldsymbol{\mu}) + \mathbf{b}_{\text{enc}})). \quad (4)$$

Here, $\mathbf{W}_{\text{skip}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is a zero-initialized, bias-free linear adapter. The sparse path projects $\mathbf{h}_t$ into an overcomplete dictionary with $d_{\text{feat}} = \alpha d_{\text{model}}$, where α is the expansion factor, and $\sigma(\cdot) = \text{ReLU}(\cdot)$ is the activation function. The Top- k operator retains the k largest activations and sets the remainder to zero, yielding a sparse vector $\mathbf{s}_t \in \mathbb{R}^{d_{\text{feat}}}$.

Table 1: Performance comparison on four grade-school mathematical reasoning benchmarks. We report averaged accuracy (Acc. %) and average latent reasoning length (#L) over five runs with different random seeds, together with 95% confidence intervals ($\pm$). Gray rows indicate ablation variants of LSTR. Among non-CoT methods, bold marks the best average accuracy, while underlining highlights a nearly matched result.

Method	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith		Average	
	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
CoT	49.4 $\pm$.72	25.6 $\pm$.11	11.9 $\pm$.16	34.2 $\pm$.11	59.8 $\pm$.29	12.1 $\pm$.03	93.2 $\pm$.49	13.7 $\pm$.09	53.6	21.4
iCoT	19.8 $\pm$.23	0.00 $\pm$.00	3.87 $\pm$.16	0.00 $\pm$.00	36.4 $\pm$.51	0.00 $\pm$.00	38.2 $\pm$.66	0.00 $\pm$.00	24.6	0.00
Coconut	23.1 $\pm$.28	6.00 $\pm$.00	5.49 $\pm$.33	6.00 $\pm$.00	40.7 $\pm$.65	6.00 $\pm$.00	41.1 $\pm$.24	6.00 $\pm$.00	27.6	6.00
Distill	13.3 $\pm$.62	6.00 $\pm$.00	2.97 $\pm$.24	6.00 $\pm$.00	21.7 $\pm$.73	6.00 $\pm$.00	19.2 $\pm$.83	6.00 $\pm$.00	14.3	6.00
CoLaR-5	26.8 $\pm$.17	5.57 $\pm$.02	5.87 $\pm$.10	6.53 $\pm$.01	48.4 $\pm$.45	2.95 $\pm$.02	86.4 $\pm$.35	3.21 $\pm$.01	41.7	4.57
LSTR-5	29.5 $\pm$.32	5.69 $\pm$.00	6.29 $\pm$.12	6.75 $\pm$.00	49.2 $\pm$.49	3.03 $\pm$.00	88.2 $\pm$.14	3.17 $\pm$.00	43.3	4.66
- w/o Skip	27.5 $\pm$.09	5.75 $\pm$.00	5.52 $\pm$.16	7.95 $\pm$.00	50.5 $\pm$.59	3.18 $\pm$.00	84.1 $\pm$.32	3.23 $\pm$.00	41.9	5.0
- w/o Sparse	20.7 $\pm$.29	5.80 $\pm$.00	4.44 $\pm$.11	6.58 $\pm$.00	41.7 $\pm$.48	3.11 $\pm$.00	75.8 $\pm$.29	3.38 $\pm$.00	35.6	4.7
CoLaR-2	40.1 $\pm$.20	12.7 $\pm$.02	9.08 $\pm$.03	14.0 $\pm$.07	54.9 $\pm$.20	6.11 $\pm$.01	91.3 $\pm$.12	7.35 $\pm$.01	48.8	10.0
LSTR-2	40.2 $\pm$.09	13.18 $\pm$.00	8.81 $\pm$.06	18.51 $\pm$.00	53.9 $\pm$.29	6.47 $\pm$.00	90.5 $\pm$.29	7.44 $\pm$.00	<u>48.4</u>	11.4
- w/o Skip	35.3 $\pm$.19	13.14 $\pm$.00	7.90 $\pm$.12	24.41 $\pm$.00	53.1 $\pm$.29	6.69 $\pm$.00	92.8 $\pm$.21	7.37 $\pm$.00	47.3	12.9
- w/o Sparse	34.2 $\pm$.22	13.43 $\pm$.00	7.32 $\pm$.17	23.37 $\pm$.00	48.6 $\pm$.36	6.49 $\pm$.00	85.3 $\pm$.37	7.57 $\pm$.00	43.9	12.7

Several design choices are intended to enhance stability under iterative latent supervision. Empirical mean subtraction with μ eliminates dominant global activation directions that would otherwise occupy sparse slots across multiple steps. Decoder columns are constrained to unit norm, $\|\mathbf{W}_{\text{dec}}^{(i)}\|_2 = 1$, to prevent feature-scale collapse. Hard Top- k selection maintains a fixed number of active features, thereby avoiding uncontrolled variations in transition capacity along the trajectory. Gradients through the non-differentiable Top- k selection are approximated using a Straight-Through Estimator, allowing selected sparse features to be optimized under hard sparsity constraints. Inactive or under-utilized features are further addressed by the feature-utilization objective detailed below.

3.4 Optimization and Feature Utilization

The complete training loss combines language-model supervision with latent transition supervision:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \beta \mathcal{L}_{\text{latent}}, \quad (5)$$

where $\mathcal{L}_{\text{LM}}$ is the cross-entropy loss applied over answer tokens and randomly sampled block-level token targets. The latent objective is formulated as:

$$\mathcal{L}_{\text{latent}} = \mathcal{L}_{\text{FVU}} + \lambda_s \mathcal{L}_{\text{skip}} + \lambda_g \mathcal{L}_{\text{ghost}}. \quad (6)$$

The primary reconstruction term utilizes the **Fraction of Variance Unexplained (FVU)** rather than raw squared error:

$$\mathcal{L}_{\text{FVU}} = \frac{\mathbb{E} \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}^*\|_2^2}{\text{Var}(\mathbf{z}_{t+1}^*)}. \quad (7)$$

This normalization accounts for variations in latent magnitude across reasoning depths and compression ratios, preventing high-magnitude targets from dominating the optimization process.

The skip objective introduces an auxiliary bias for the skip path:

$$\mathcal{L}_{\text{skip}} = \mathbb{E} \|(\mathbf{W}_{\text{skip}} \mathbf{h}_t + \mathbf{b}_{\text{dec}}) - \mathbf{z}_{t+1}^*\|_2^2. \quad (8)$$

Because the skip path is linear and bias-free, it is ideally suited to capture predictable components of consecutive latent states. This reduces the representational burden on the sparse path, whose active capacity is inherently restricted by the Top- k selection.

Under hard sparsity, certain encoder units may rarely enter the active set. To improve dictionary utilization, LSTR incorporates a ghost-gradient objective that propagates reconstruction residuals to inactive or under-utilized encoder features. Let $\mathbf{r}_t = \mathbf{z}_{t+1}^* - \hat{\mathbf{z}}_{t+1}$ denote the transition residual. The ghost objective is defined as:

$$\mathcal{L}_{\text{ghost}} = \mathbb{E} \|\mathbf{W}_{\text{dec enc}}(\mathbf{h}_t)_{\text{dead}} - \mathbf{r}_t\|_2^2. \quad (9)$$

This auxiliary term encourages unused dictionary features to align with unexplained residual directions, thereby mitigating feature collapse during training.

4 Experiments

Our evaluation of LSTR begins by testing whether sparse transitions maintain the trade-off between accuracy and trajectory length in latent reasoning

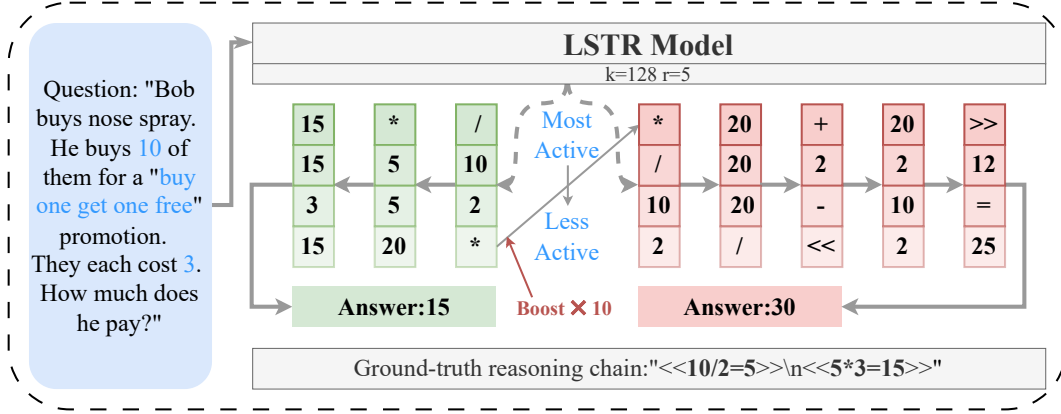


Figure 3: A qualitative illustration of how perturbing a single active feature can alter the latent trajectory. Systematic semantic annotation of sparse features is left to future work.

Table 2: Inference-time path masking on GSM8K. All rows use the same trained LSTR model.

Masked path	Remaining paths	Acc. (%)	Δ
None	Skip + Sparse	40.2	–
Sparse	Skip only	35.2	-5.0
Skip	Sparse only	0.3	-39.9

under matched compression. We then examine the model’s internal dynamics during inference, specifically the functional roles of the two LTT pathways and the active sparse features. We also analyze how the sparsity budget controls transition capacity and whether these behaviors generalize beyond arithmetic tasks. Throughout the experiments, explicit CoT serves as an uncompressed reference, while dense latent reasoning methods, particularly CoLaR, act as the primary baseline under identical compression conditions.

4.1 Experimental Setup

Datasets and metrics. We primarily train and evaluate our models on **GSM8k-Aug** (Deng et al., 2023), an augmented version of GSM8k containing approximately 385k training samples and 1k test samples. To assess generalization across reasoning distributions, we additionally evaluate on three out-of-domain benchmarks: **GSM-Hard** (Gao et al., 2023), **SVAMP** (Patel et al., 2021), and **MultiArith** (Roy and Roth, 2016). For evaluating scalability to more challenging reasoning regimes, we further report results on the **MATH** dataset (Hendrycks et al., 2021). Following prior work (Hao et al., 2024), we report **Accuracy (Acc.)** as the percentage of correctly solved problems and

Reasoning Length (#L), measured as the average number of generated reasoning tokens or latent transitions, as a proxy for inference cost.

Baselines. We compare LSTR against the following representative baselines: (1) **CoT** (Wei et al., 2022), fine-tuned on complete reasoning chains, performs token-by-token reasoning before outputting answers; (2) **iCoT** (Deng et al., 2024), gradually internalizes reasoning steps and directly generates answers without intermediate tokens; (3) **Coconut** (Hao et al., 2024), progressively replaces token reasoning with latent steps through curriculum learning, executing fixed-length latent reasoning; (4) **Distill** (Shen et al., 2025), distills token-level CoT into latent reasoning steps and infers similarly to Coconut. (5) **CoLaR** (Tan et al., 2025), performs dynamic latent reasoning with compression, trained for speed-adjustable inference.

Implementation details. All models use the Llama-3.2-1B-Instruct backbone (Team, 2024). We freeze the backbone parameters and employ **LoRA** for parameter-efficient fine-tuning. Following (Tan et al., 2025), each model is trained for at most 50 epochs or 12 hours. The checkpoint with the best validation accuracy is used for evaluation. For LSTR, the feature expansion factor is set to 16, and the default sparsity budget is $k = 128$ active features per latent step. Unless otherwise stated, results are averaged over five random seeds. Appendix A gives the full hyperparameters, optimizer settings, and hardware details.

4.2 Performance under Latent Compression

Table 1 presents the main results across four mathematical reasoning benchmarks. The objective of this comparison is not to demonstrate that latent reasoning achieves higher accuracy than explicit CoT. Explicit CoT remains the strongest token-level reference across several benchmarks, despite requiring substantially longer reasoning chains. Instead, the primary comparison focuses on dense latent reasoning methods under identical compression settings, particularly CoLaR.

At strong compression ($r = 5$), LSTR outperforms CoLaR on GSM8K-Aug and GSM-Hard while performing comparably on SVAMP and MultiArith, increasing average accuracy from 41.7% to 43.3% at similar latent lengths. At milder compression ($r = 2$), LSTR averages 48.4% accuracy versus 48.8% for CoLaR, demonstrating sparse transitions competitively replace dense ones without compromising the accuracy-length trade-off (Appendix B details sensitivity for $r \in \{1, 2, 3, 4, 5\}$). Train-time ablations (Table 1, gray rows) reveal both pathways are essential when training from scratch, complementing Section 4.3. Eliminating the sparse path degrades latent transition modeling, dropping accuracy from 43.3% to 35.6% at $r = 5$ and 48.4% to 43.9% at $r = 2$. Removing the skip path similarly reduces performance and extends trajectory length at $r = 2$ from 11.4 to 12.9, confirming its stabilizing transport function.

4.3 Causal Analysis of Sparse Transitions

We next investigate whether LSTR actively uses sparse transitions during inference, beyond merely altering model parameterization. We apply two interventions: masking both LTT paths in the same trained model to isolate their roles, and removing active sparse features to evaluate their impact on final reasoning outcomes.

Inference-time path masking. The LTT consists of a skip path and a sparse path. To isolate their respective effects, we train the complete LSTR model once and mask one of the pathways during inference. Table 2 presents the evaluation results on GSM8K-Aug. The two pathways exhibit distinct failure modes. Removing the sparse path yields a skip-only model with an accuracy of 35.2%, suggesting that the linear path sustains a substantial portion of the continuous latent trajectory. The performance gap of 4.9% relative to the full model demonstrates that the sparse path provides criti-

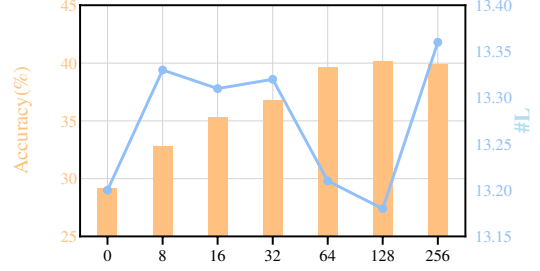


Figure 4: Accuracy and average latent reasoning length under different inference-time sparsity budgets k . The model is trained with $r = 2$ and $k = 128$, and evaluated by changing k only at inference time.

cal residual updates beyond linear transport. Conversely, ablating the skip path reduces the accuracy to 0.3%, indicating that sparse updates alone cannot maintain the latent trajectory in the absence of a stable transport channel. These findings highlight a clear functional division: the skip path preserves the continuity of the latent state, whereas the sparse path provides residual transition updates that enhance reasoning capabilities.

Feature-level causal ablation. To evaluate whether active sparse features affect accuracy, we ablate the N most active features per latent step and measure the accuracy change, comparing against a baseline of N randomly ablated active features. For CoLaR, we identically ablate the N latent dimensions with the largest absolute activations (Table 3). Ablating the top-active features in LSTR steadily degrades accuracy, dropping 16.5 percentage points at $N = 10$, whereas ablating random active features yields a substantially smaller effect. Similar intervention on the top dense dimensions of CoLaR induces a smaller drop, confirming that computation is more distributed across dense latent states. While not implying every sparse feature maps to a fully interpretable concept, these results substantiate that active sparse features in LSTR causally participate in the latent transition process.

Qualitative feature intervention. For visualization, we assign feature labels by projecting sparse features onto the token embedding space with cosine similarity. This projection is only a weak semantic probe that helps make trajectories readable. The causal claim is based on Table 3. Figure 3 shows one intervention example, where modifying an early active feature changes the later trajectory and the final answer. Additional structural analyses are provided in Appendix E.

Table 3: Feature-level causal ablation on GSM8K-Aug. Accuracy change (percentage points) after removing N sparse features or dense latent dimensions during inference. Blue and gray rows denote top-ranked and random units, respectively.

Model	$N = 1$	$N = 3$	$N = 5$	$N = 10$
LSTR	-4.6	-11.8	-15.1	-16.5
CoLaR	+1.0	+0.7	-0.7	-3.0
LSTR	-0.3	-0.7	-2.0	-0.7
CoLaR	+1.3	-1.0	-1.0	+2.0

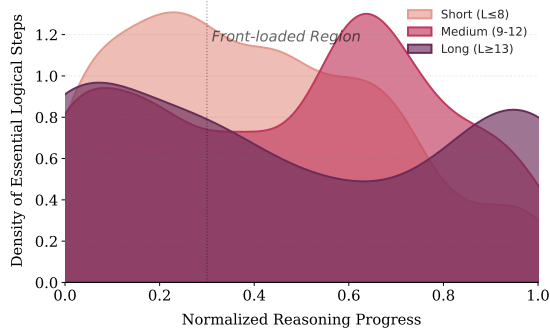


Figure 5: KDE-estimated temporal density of causally necessary latent steps ($r = 2$) across heterogeneous trajectory lengths ($L \in [4, 35]$). Length-balanced cohorts exhibit a consistent front-loaded profile.

4.4 Capacity Control and Sparse Dynamics

We study how sparse transition capacity is controlled and used across latent reasoning trajectories. The sparsity budget k sets the maximum number of active features at each latent step, allowing inference-time capacity adjustments without retraining. Figure 4 shows that accuracy improves when more sparse features participate, while the average latent reasoning length remains relatively stable. This result indicates that k directly controls the capacity of sparse latent transitions, rather than serving only as a post-hoc analysis parameter. This control should not be interpreted as semantic control over the model, since changing k adjusts how many sparse features contribute to reasoning without determining which semantic operation the model performs. Appendix B reports a separate compression-ratio analysis, where r controls latent trajectory length rather than the number of active features per transition.

We then examine where causal necessity manifests within compressed latent trajectories. Figure 5 estimates the distribution of causally necessary latent steps under $r = 2$ across trajectories of different lengths. The density is consistently front-

loaded across length-balanced cohorts, suggesting that early latent transitions carry a larger share of task-relevant updates. This pattern agrees with the feature-level ablation in Table 3: sparse features are causally relevant in aggregate, and their influence is organized over time.

4.5 Generalization and Efficiency

The main experiments focus on mathematical reasoning, where reasoning steps are closely tied to numbers and operators. To test whether sparse latent transitions also apply outside arithmetic, we evaluate LSTR on StrategyQA (Geva et al., 2021), a commonsense reasoning benchmark. Under the same compression setting, LSTR achieves 61.13% accuracy, compared with 58.49% for CoLaR and 64.83% for explicit CoT. This result suggests that sparse latent transitions are not limited to arithmetic reasoning, although evaluation on more natural-language reasoning tasks is still needed.

Appendix C analyzes computational overhead. The sparse transition operator adds per-step computation because it uses an expanded dictionary and Top- k selection. On GSM8K-Aug, LSTR ($r = 2$) has higher per-step latency than CoLaR ($r = 2$), but it reduces end-to-end time relative to explicit CoT by using a shorter reasoning trajectory. The efficiency gain of LSTR therefore comes mainly from trajectory compression, not from faster individual latent transitions. Appendix D reports results on harder tasks and a larger backbone, where the accuracy gap between latent reasoning and explicit CoT remains a major limitation on challenging mathematical benchmarks.

5 Conclusion

We propose LSTR, advancing sparse transcoders from post-hoc diagnostics to direct transition operators in latent reasoning. It computes transitions using bounded Top- k sparse features and a linear skip path for predictable drift, matching dense baselines under matched compression while exposing feature-level trajectory access. Causal ablations show that removing top-active features substantially degrades the model but only weakly affects dense baselines, indicating these features causally participate in transitions. Although a gap with explicit Chain-of-Thought persists, LSTR is a step toward causally testable reasoning. Future work includes reducing CoT supervision and developing stronger interpretation tools.

Limitations

LSTR remains subject to several limitations. First, it relies on compressed targets derived from explicit Chain-of-Thought traces, so it currently transfers existing reasoning structures into latent space rather than learning them without CoT supervision. Second, latent compression still trails explicit CoT on harder symbolic benchmarks such as MATH, where fine-grained intermediate states are important for long-horizon compositional reasoning. Third, the interpretability of LSTR is local: feature-level interventions reveal causally active transition components, but they do not provide a complete symbolic reconstruction of the full reasoning trajectory. Finally, although we include StrategyQA as an initial non-arithmetic benchmark, most mechanistic analyses are still performed on mathematical tasks. Extending the evaluation to broader natural-language reasoning and agentic settings remains future work.

Ethics Statement

This work utilizes publicly available reasoning benchmarks and pretrained language models, without involving new data collection, human-subject experiments, or the processing of personally identifiable information. The proposed sparse-transition interface is designed to support the debugging, mechanistic analysis, and auditing of latent reasoning. However, this capability for feature-level intervention may also facilitate more precise activation steering if subjected to misuse. Therefore, we present these interventions strictly for controlled analysis and emphasize that the application of such techniques in safety-critical or user-facing systems requires additional safeguards and continuous monitoring.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62506166, U2441285), the Natural Science Foundation of Jiangsu Province (No. BK20251365), the China Postdoctoral Science Foundation (No. 2025M774283), the Scientific Research Starting Foundation of Nanjing University of Aeronautics and Astronautics (No. 1015-YAH24096), and the Open Project of the State Key Laboratory of Ocean Sensing (No. OSKF-2025M10). This work was sponsored by the Chinese Association for Artificial Intelligence - Huawei AI Computing Power

Acceleration Program. This research was also supported by the DiDi GAIA Collaborative Research Funds (No. CCF-DiDi GAIA202507) and CAAI-MindSpore Open Fund (CAAIXSJLJJ 2025 MindSpore 01), developed on OpenI Community.

References

- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, and 5 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. <https://transformer-circuits.pub/2023/monosemantic-features/index.html>.
- Xiang Chen, Yixin Ou, Quan Feng, Lei Li, Piji Li, Haibo Ye, Sheng-Jun Huang, Shuofei Qiao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2025a. Retrieval-augmented prompt learning for pre-trained foundation models. *IEEE Transactions on Audio, Speech and Language Processing*, 33:4311–4322.
- Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2778–2788. ACM.
- Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025b. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 28241–28259. Association for Computational Linguistics.
- DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.
- Yuntian Deng, Yejin Choi, and Stuart M. Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *CoRR*, abs/2405.14838.
- Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart M. Shieber. 2023. Implicit chain of thought reasoning via knowledge distillation. *CoRR*, abs/2311.01460.
- Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. 2024. Transcoders find interpretable LLM feature circuits. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [PAL: program-aided language models](#). In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies](#). *Trans. Assoc. Comput. Linguistics*, 9:346–361.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. [Training large language models to reason in a continuous latent space](#). *CoRR*, abs/2412.06769.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helvar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, and 80 others. 2024. [Openai o1 system card](#). *CoRR*, abs/2412.16720.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094. Association for Computational Linguistics.
- Gonalo Paulo, Stepan Shabalin, and Nora Belrose. 2025. [Transcoders beat sparse autoencoders for interpretability](#). *CoRR*, abs/2501.18823.
- Subhro Roy and Dan Roth. 2016. [Solving general arithmetic word problems](#). *CoRR*, abs/1608.01413.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. [Reasoning with latent thoughts: On the power of looped transformers](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. [CODI: compressing chain-of-thought into continuous space via self-distillation](#). *CoRR*, abs/2502.21074.
- Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. [Think silently, think fast: Dynamic latent compression of LLM reasoning chains](#). *CoRR*, abs/2505.16552.
- Llama Team. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Chenxi Wang, Xiang Chen, Ningyu Zhang, Bozhong Tian, Haoming Xu, Shumin Deng, and Huajun Chen. 2025. [MLLM can see? dynamic correction decoding for hallucination mitigation](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. 2025. [Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought](#). *CoRR*, abs/2501.04682.
- Ruijie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. [A survey on latent reasoning](#). *CoRR*, abs/2507.06203.

A Implementation Details

Model Hyperparameters All experiments are conducted using the Llama-3.2-1B-Instruct backbone (Team, 2024). The backbone parameters are frozen throughout training and adapted using LoRA, while the LTT parameters are trained directly. We set the LoRA rank to $r_{\text{LoRA}} = 128$ and the scaling factor to $\alpha_{\text{LoRA}} = 32$ for all experiments. The latent reasoning module adopts the LTT architecture defined in Section 3.3, where the latent feature dimension is set to $d_{\text{feat}} = 16 \times d_{\text{model}}$. Unless otherwise specified, the sparsity budget is fixed to $k = 128$ active features per latent step. The LTT encoder is initialized with orthogonal weights, while decoder columns are constrained to unit norm to stabilize feature scaling. The decoder bias is initialized using the empirical mean hidden state computed over the training corpus. The skip path linear adapter is initialized to zero, ensuring that non-linear transition residuals are initially routed through the sparse path. All latent activations use ReLU nonlinearity, and the latent temperature is fixed to 1.0.

Training and Optimization Protocol All models are trained exclusively using **Supervised Fine-Tuning (SFT)**; we do not employ reinforcement learning or policy optimization in any experiment. We utilize the AdamW optimizer with a base learning rate of 1×10^{-4} and weight decay of 1×10^{-2} . To accommodate the different convergence properties of the backbone and the sparse transcoder, we implement a hierarchical learning rate strategy:

The sparsity budget is fixed at $k = 128$ throughout training. Gradient clipping with a maximum norm of 1.0 is applied to stabilize training. All experiments are conducted on **two NVIDIA RTX PRO6000 GPUs** using mixed-precision training. Unless otherwise stated, results are averaged over five random seeds (0–4).

B Sensitivity Analysis of Compression Ratio r

In the main paper, we fix the sparsity budget to $k = 128$ and focus on comparing established latent compression ratios. In this appendix, we provide supplementary results analyzing how the compression strength r affects reasoning performance and trajectory compactness. Specifically, we evaluate LSTR models across a range of compression ratios $r \in \{1, 2, 3, 4, 5\}$, while maintaining a fixed

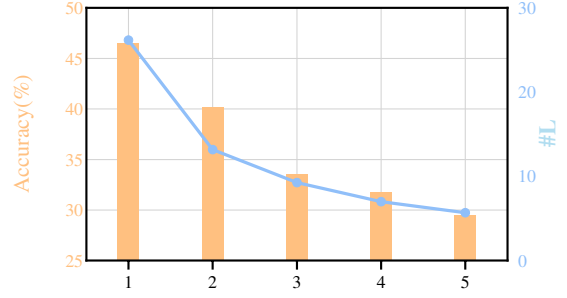


Figure 6: **Accuracy and reasoning length (#L) across varying compression ratios r .** The orange bars denote reasoning accuracy (%), while the blue line represents the average number of latent steps (#L) per problem. Increasing r leads to significantly more compact trajectories but results in a corresponding decline in accuracy.

sparsity budget of $k = 128$ and keeping all other hyperparameters constant. Figure 6 illustrates the corresponding accuracy and average latent reasoning length. As the compression ratio r increases from 1 to 5, the latent trajectory becomes progressively shorter, resulting in substantially and monotonically reduced reasoning length. However, aggressive compression also leads to a gradual degradation in accuracy, reflecting a reduced capacity of the latent manifold to preserve fine-grained reasoning signals under aggressive compression. These results demonstrate a clear and smooth accuracy–efficiency Pareto frontier controlled by the compression ratio r , confirming that LSTR enables controllable latent reasoning compression without requiring architectural changes or additional model retraining.

C Training and Inference Overhead

We analyze the computational overhead of LSTR relative to explicit CoT and the dense *CoLaR* baseline. All measurements were conducted on a single NVIDIA RTX PRO6000 GPU using FP16 precision with a batch size of 1.

Inference Latency. As shown in Table 4, LSTR introduces additional computation at each latent step due to the $16\times$ feature expansion in the LTT encoder and the Top- k selection process. Specifically, the **1.02 ms** required for the LTT operation results in a per-step latency of **10.66 ms**, representing a **1.68 ms** increase over the dense baseline (8.98 ms). However, this per-step increase is effectively offset by the reduction in total reasoning length. Compared to explicit CoT, LSTR-2 reduces

Parameter Group	Learning Rate	Weight Decay	Functional Role
Backbone (LoRA)	1×10^{-4}	0.1	Backbone adaptation
Skip Path ($\mathbf{W}_{\text{skip}}$)	2×10^{-4}	0.0	Skip path
LTT Encoder	2×10^{-4}	0.0	Sparse path encoder
LTT Decoder	2×10^{-4}	0.0	Sparse path decoder

Table 4: Efficiency profiling on GSM8k-Aug. Per-step latency and Peak VRAM were measured during inference. End-to-end (E2E) time is calculated as Average Steps $\times$ Step Latency.

Method	Acc. (%)	Avg. Steps (#L)	Step Latency	Policy Overhead	Peak VRAM
CoT (Explicit)	49.4	25.60	9.26 ms	—	5.14 GB
CoLaR-2 (Dense)	40.1	12.70	8.98 ms	1.69%	5.21 GB
LSTR-2 (Ours)	40.2	13.18	10.66 ms	10.53%	5.66 GB

the reasoning sequence from **25.6** tokens to **13.18** steps. Consequently, the total end-to-end (E2E) inference time for a representative reasoning chain is reduced from **237.1 ms** (CoT) to **140.5 ms** (LSTR-2), achieving a **40.7%** speed improvement. This indicates that LSTR improves efficiency primarily by significantly reducing reasoning length rather than minimizing per-step computation.

Memory and Training. During training, the backbone remains frozen, and overhead arises primarily from sparse feature selection and latent trajectory supervision. The introduction of **138.45M** parameters in the LTT module increases the peak VRAM usage by approximately **450 MB** compared to the dense model (5.66 GB vs. 5.21 GB) and **520 MB** compared to the CoT baseline. While the static memory footprint is higher, the significantly shorter latent trajectories lead to a slower growth of the **KV Cache** during generation. This makes sparse latent reasoning a practical and scalable alternative to explicit CoT, as the benefits of trajectory compression outweigh the marginal increases in per-step latency and static parameter storage.

D Cross-Backbone Consistency

To evaluate whether sparse latent reasoning extends beyond small-scale backbones, we conduct scalability experiments with LSTR on DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025) and Llama-3.2-8B-Instruct (Team, 2024). All experiments are performed on the MATH benchmark, which constitutes a substantially more challenging reasoning setting due to its complex symbolic dependencies and multi-step deduction requirements. For all scalability evaluations, LSTR adopts a fixed configuration with a compression ratio of $r = 2$ and a sparsity budget of

$k = 128$. This setting follows the training protocol introduced in Section 4.1 and enables a direct comparison with the CoLaR baseline under the same compression ratio.

Across both DeepSeek-based and Llama-based backbones, LSTR consistently produces substantially shorter reasoning trajectories than explicit CoT decoding. The compression advantage remains stable as model scale increases, indicating that sparse latent transitions can internalize nontrivial reasoning procedures even within high-dimensional representation spaces. Although a performance gap relative to CoT remains, the gap is expected given the difficulty of preserving fine-grained symbolic reasoning within a fully latent reasoning process. Importantly, the overall behavior of LSTR remains highly consistent across the two model families. Both accuracy and reasoning-length trends exhibit similar patterns for the DeepSeek and Llama backbones despite differences in architecture design and distillation strategies. This stability across heterogeneous backbones suggests that the effectiveness of sparse latent reasoning primarily originates from the LTT formulation rather than from backbone-specific properties. Overall, the results demonstrate that LSTR provides a scalable and structurally consistent framework for efficient reasoning in LLMs while preserving the efficiency-interpretability trade-off under increasing model capacity.

E Latent Structural Analysis

To comprehensively understand the internal dynamics of sparse latent reasoning, we analyze the structural properties of LSTR at both the micro level of individual feature activations and the macro level

Table 5: Scalability evaluation on 7B+ backbones over the MATH benchmark. All models use a compression ratio of $r = 2$ and a sparsity budget of $k = 128$. Across different model families, LSTR preserves a stable efficiency–accuracy trade-off, indicating that the sparse transition formulation can be applied across model families, while hard symbolic reasoning remains challenging under latent compression.

Method	DeepSeek-R1-Distill-Qwen-7B		Llama-3.2-8B-Instruct	
	Acc.	# L	Acc.	# L
CoT	35.2 $\pm$.31	198.0 $\pm$ 1.2	24.1 $\pm$.27	200.0 $\pm$ 1.5
CoLaR	16.5 $\pm$.14	81.2 $\pm$.23	8.0 $\pm$.19	85.4 $\pm$.44
LSTR	15.9 $\pm$.11	72.9 $\pm$.00	10.9 $\pm$.23	82.9 $\pm$.01

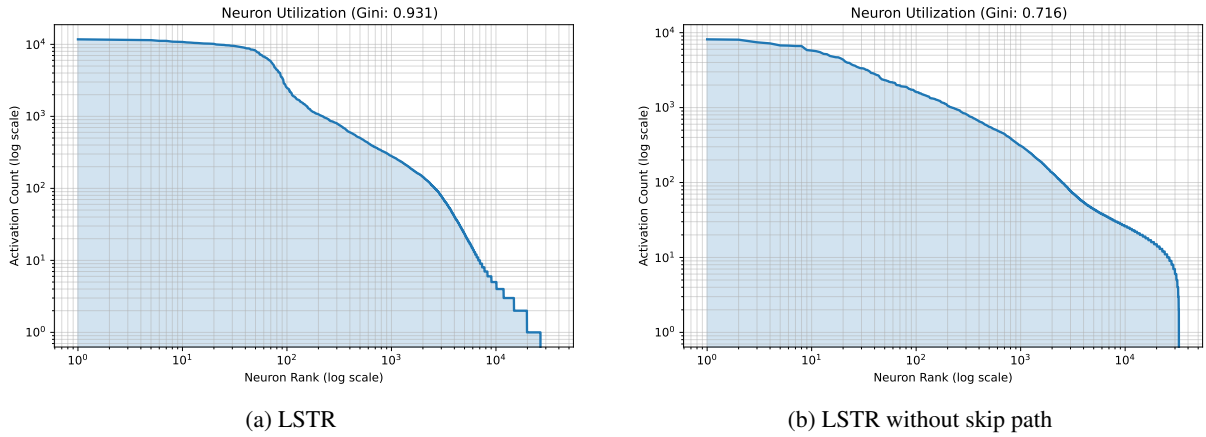


Figure 7: Feature utilization under Top- k inference. The full model shows a more concentrated utilization pattern, while the model without the skip path uses features more diffusely.

of global trajectory evolution.

Feature Utilization At the micro level, we analyze the utilization frequency of LTT features by counting their activations under Top- k inference. As shown in Figure 7, the resulting rank–frequency distribution follows a clear power-law pattern. In the full LSTR architecture, feature utilization is highly concentrated, yielding a Gini coefficient of approximately 0.93, which indicates the strong specialization of a small subset of latent features. In contrast, when the skip path is removed and only the sparse path is retained, feature utilization becomes significantly more diffuse, reducing the Gini coefficient to 0.71. This comparison highlights the crucial functional role of the skip path: by absorbing predictable background drift, it allows sparse features to highly specialize in discrete semantic innovations.

Latent Trajectory Coherence Given the highly specialized and sparse nature of these individual updates, a natural question is whether such hard representational bottlenecks disrupt the overall continuity of the reasoning process. To address this,

we examine the macro-level semantic evolution of latent embeddings across successive steps. Figure 9 illustrates the cosine similarity matrix between reconstructed latent embeddings within a single reasoning trajectory. Adjacent latent steps exhibit high similarity, which preserves local temporal continuity. Meanwhile, as illustrated in Figure 8, the similarity between the initial and final steps is substantially lower (Start–End similarity: 0.339), indicating a smooth yet progressive semantic drift. This pattern confirms that latent reasoning in LSTR follows a coherent, goal-directed semantic trajectory, successfully avoiding both the collapse into static representations and the instability of random fluctuations.

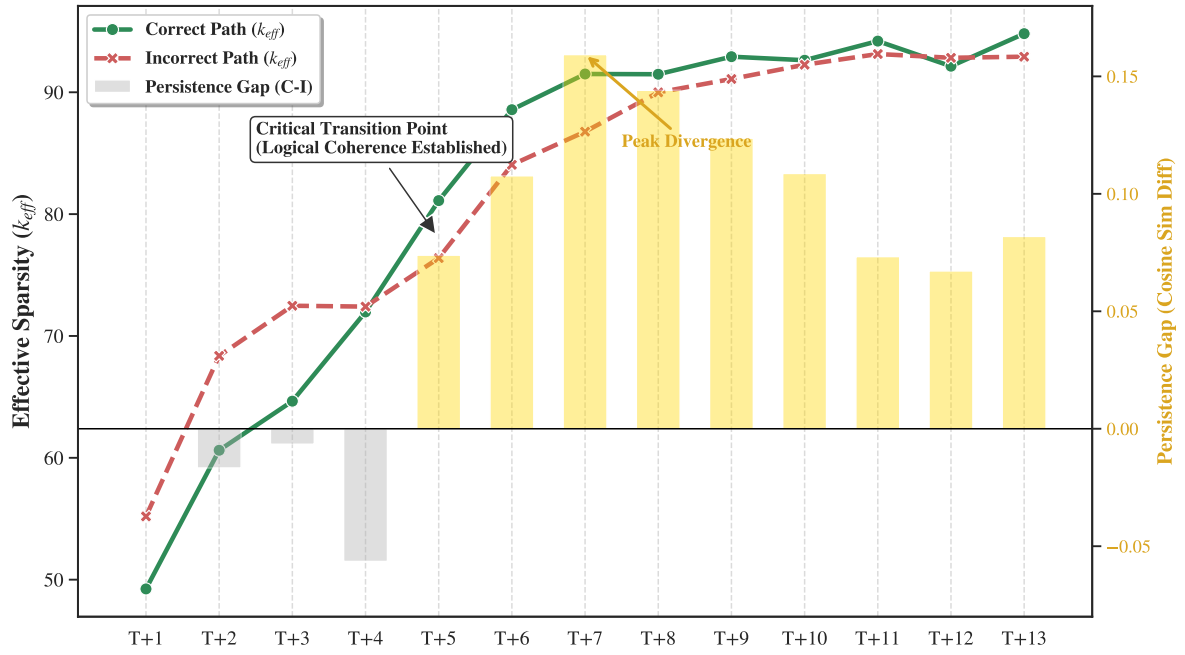


Figure 8: Step-wise effective sparsity k_{eff} and feature persistence along latent reasoning trajectories. The model is trained with $r = 2$ and $k = 128$; sparsity is fixed at inference time. Correct and incorrect trajectories are aggregated separately.

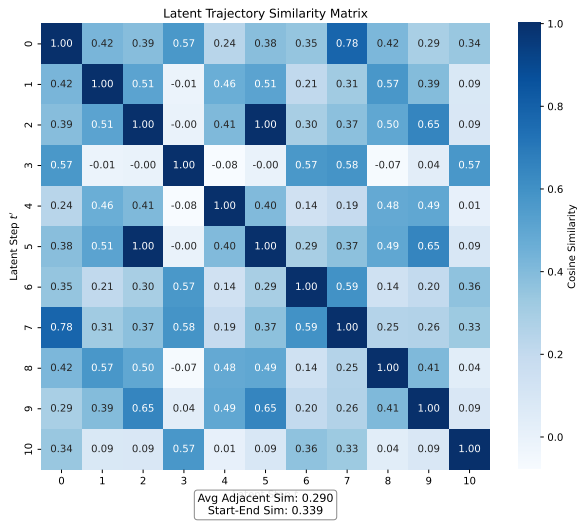


Figure 9: Cosine similarity between reconstructed latent embeddings across latent steps. High similarity near the diagonal indicates local temporal continuity, while lower similarity between early and late steps reflects gradual latent-state change.