SINGULAR VALUE PERTURBATION: UNLOCKING RL PERFORMANCE VIA DIVERSE LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent Reasoning Models (LRMs) replace explicit Chain-of-Thought tokens with continuous recurrent computation, yet their deterministic recurrence limits the exploration needed for effective reinforcement learning. Sampling answer tokens alone cannot explore alternative computations within the latent reasoning phase. We introduce **Singular-Value Perturbation (SVP)**, which diversifies latent trajectories by randomly rescaling coefficients in the fixed SVD basis of a weight matrix. Applied to attention Value projections as **SVP-V**, it perturbs the messages transmitted by attention while preserving the attention weights for a fixed input. Across multiple LRMs and mathematical reasoning benchmarks, SVP-based configurations achieve the highest solution coverage in all evaluated settings, while intermediate LM-head readouts reveal semantic branching around decision-critical steps. Building on this exploration mechanism, **SVP-V-GRPO** learns from the relative outcomes of perturbed trajectories while updating only the Value parameters. Starting from COCONUT, it improves GSM8K accuracy from 34.1% to 50.3%, achieving state-of-the-art performance across all six benchmarks among GPT2-based continuous LRMs. These results establish structured latent exploration as an effective mechanism for RL post-training of continuous LRMs.

1 INTRODUCTION

Latent Reasoning Models (LRMs) replace explicit Chain-of-Thought tokens with continuous recurrent computation, substantially reducing decoding and context overhead (Hao et al., 2024; Shen et al., 2025; Wei et al., 2025). However, their latent recurrence is typically deterministic, causing repeated runs follow the same latent trajectory. This limits exploration during reinforcement learning: sampling answer tokens can diversify outputs but cannot alter the latent computations that produce them. The problem is especially relevant to Group Relative Policy Optimization (GRPO) (Shao et al., 2024), where a group of uniformly successful or unsuccessful rollouts have zero centered advantage. Exploring alternative latent computations can uncover successful trajectories and provide the reward contrast needed for learning.

We introduce **Singular-Value Perturbation (SVP)**, a structured exploration mechanism that randomly rescales rank-one components in the fixed SVD basis of a pretrained weight matrix. Unlike unstructured noise, SVP varies the contributions of learned transformation directions while preserving their singular vectors. Applied to attention Value projections as **SVP-V**, it perturbs the messages transmitted through the current attention weights without directly modifying Query–Key routing at the point of perturbation, driving subsequent hidden states along distinct trajectories. Within this fixed-SVD class, we show that independent Gaussian coefficient perturbations uniquely maximize joint differential entropy under a fixed response-relative perturbation budget.

Across multiple LRMs and six mathematical reasoning benchmarks, SVP-based configurations achieve the highest solution coverage in every evaluated setting, with SVP-V providing the strongest overall performance as a single configuration. SVP-V consistently outperforms answer-token sampling and Gaussian activation noise. LM-head readouts further reveal semantically distinct branches around decision-critical steps, including alternative intermediate computations that recover correct solutions missed by the deterministic trajectory.

Finally, we introduce **SVP-V-GRPO**, which learns from the relative outcomes of perturbed latent trajectories while updating only the attention Value parameters. Starting from COCONUT-GPT2,

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

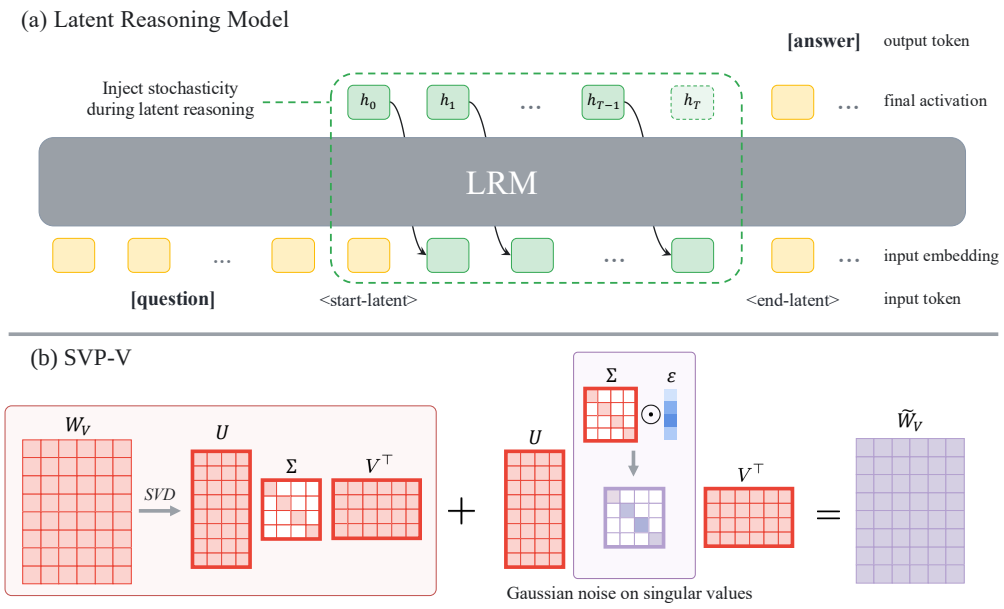

Figure 1: **Overview of Latent Reasoning Models (LRMs) and SVP-V. (a)** LRMs recurrently update continuous hidden states across latent steps before explicit answer decoding. **(b)** SVP-V decomposes the attention Value weight and perturbs the coefficients of rank-one SVD components.

SVP-V-GRPO improves GSM8K accuracy from 34.1% to 50.3% under clean greedy decoding with perturbations disabled. It improves the starting checkpoint on all six benchmarks and achieves the best performance among the evaluated GPT-2-based continuous LRMs on every benchmark. These results demonstrate that structured latent exploration can translate broader solution coverage into substantial improvements in the unperturbed model.

Our contributions are:

- **Structured latent-trajectory sampling.** We introduce SVP, which perturbs the coefficients of rank-one SVD components while preserving their left and right singular vectors, and instantiate it on attention Value projections as SVP-V.
- **Broader and semantically meaningful solution exploration.** SVP-based configurations achieve the highest solution coverage in all main settings, with SVP-V providing the strongest individual configuration. LM-head readouts indicate semantically diverse reasoning branches around decision-critical steps.
- **Effective RL post-training through latent exploration.** We introduce SVP-V-GRPO, which learns from perturbed latent trajectories while updating only the Value parameters. It substantially improves clean greedy accuracy and achieves state-of-the-art performance across all six benchmarks among the evaluated GPT-2-based continuous LRMs.

1.1 Related Work

Prior work explores internal reasoning through token-wise rationales (Quiet-STaR) (Zelikman et al., 2024), pause tokens (Goyal et al., 2024), and latent planning (PLaT) (Wang et al., 2026a). Beyond scaling decoded reasoning traces (Snell et al., 2024; Muennighoff et al., 2025), latent exploration uses activation noise or dropout (You et al., 2025; Wang et al., 2025), learned samplers, or diffusion-based methods (Wang et al., 2026b; Tan et al., 2025; Kang et al., 2026). TokUR instead uses low-rank weight perturbations for token-level uncertainty estimation (Zhang et al., 2025). In contrast, SVP-V perturbs coefficients in the fixed learned SVD basis of Value projections to diversify pre-trained latent trajectories without auxiliary samplers.

Recent work also applies policy optimization to stochastic latent reasoning. LEPO (Zhou et al., 2026) and Latent-GRPO (Deng et al., 2026) inject Gumbel noise into vocabulary-space mixtures at each latent step, which are unavailable in hidden-state-feedback LRMs such as COCONUT and CoDi. Parameter-space alternatives include rollout-wise Bernoulli masks (Jung, 2026). SVP-V-GRPO instead perturbs spectrum-scaled coefficients in the fixed SVD basis of W_V, preserving Query–Key routing at the point of injection and aligning the exploration distribution with the trainable subspace.

2 PRELIMINARIES

2.1 LATENT REASONING AND PARALLEL SAMPLING

Latent Reasoning Models (LRMs) such as COCONUT (Hao et al., 2024), CoDi (Shen et al., 2025), and SIM-CoT (Wei et al., 2025) perform reasoning through recurrent continuous states, or latent reasoning steps, as illustrated in Figure 1(a). Given a prompt sequence w, the model first processes w during prefill to obtain an initial hidden state h_0. Next, it iteratively updates the hidden state for T steps. Finally, the model generates the answer explicitly in tokens from the final hidden state h_T. Let F_θ denote the latent-state update function parameterized by θ. The resulting trajectory is

$$h_{t+1} = F_\theta(h_t), \qquad t = 0, \ldots, T-1, \qquad \tau_\theta(w) = (h_0, \ldots, h_T). \tag{1}$$

For simplicity, this notation omits other dependencies of F_θ such as the prompt context and cached attention keys and values. Note that F_θ is deterministic at inference time, hence a given prompt w always yields the same trajectory $\tau_\theta(w)$. Although sampling answer tokens can produce different final answers, it does not change the latent computation that produced h_T.

Parallel sampling runs the model multiple times on the same input to generate multiple candidate solutions. We measure the resulting solution coverage using pass@n, the probability that at least one of n sampled solutions is correct. Given c correct samples among $N \geq n$ i.i.d. samples, an unbiased estimator of pass@n is $1 - \binom{N-c}{n} / \binom{N}{n}$ (Chen et al., 2021). This metric requires no particular answer-selection rule. However, with deterministic latent reasoning and greedy decoding, all n runs are identical, so pass@n reduces to pass@1.

2.2 ATTENTION IN TRANSFORMER-BASED LRMS

The LRMs considered in this work parameterize F_θ as a stack of L Transformer blocks (Vaswani et al., 2017), each block containing a multi-head self-attention module and an MLP. At each latent reasoning step, the current latent representation is processed through the stack, with self-attention providing access to the prompt and preceding latent states.

We adopt the row-vector convention, so a linear layer with weight W maps an input x to xW. Let $X \in \mathbb{R}^{T \times d_{\text{model}}}$ be the matrix of input representations over T positions. In a multi-head self-attention module with H heads and per-head Query–Key dimension d_k, each head $h \in \{1, \ldots, H\}$ is parameterized by Query and Key projections $W_{Q,h}, W_{K,h} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, a Value projection $W_{V,h} \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and a corresponding slice of the output projection $W_{O,h} \in \mathbb{R}^{d_v \times d_{\text{model}}}$. The contribution of head h to the attention output is:

$$\begin{aligned} P_h(X) &= \text{softmax}\left(\frac{(XW_{Q,h})(XW_{K,h})^\top}{\sqrt{d_k}} \right), \\ Y_h(X) &= \underbrace{P_h(X)}_{\text{attention matrix}} \underbrace{(XW_{V,h})}_{\text{Value vectors}} \underbrace{W_{O,h}}_{\text{output projection}}, \end{aligned} \tag{2}$$

where the softmax is applied row-wise over the positions visible to each query under the causal mask. The total multi-head attention output is the sum over all heads, $Y(X) = \sum_{h=1}^{H} Y_h(X)$, equivalently obtained by concatenating the head outputs and applying the full output projection W_O.

Following the QK/OV circuit perspective of Elhage et al. (2021), we distinguish two computations within an attention head. The *QK circuit* uses the Query and Key projections to determine the attention matrix $P_h(X)$, where $(P_h)_{ij}$ specifies how strongly query position i attends to source position j. We refer to this weighting across source positions as *attention routing*, consistent with

the message-routing view of self-attention (Sanford et al., 2024). The *OV circuit*, in contrast, uses the Value and output projections to transform each source representation into a message. Under our row-vector convention, this *message transformation* is parameterized by $W_{V,h}W_{O,h}$. For each query position, the messages are weighted by the corresponding row of $P_h(X)$ and summed to produce the head's contribution to the residual stream. Thus, the QK circuit determines how strongly each source contributes, while the OV circuit determines what vector each source contributes. Unless stated otherwise, we use $W_V = [W_{V,1}, \ldots, W_{V,H}] \in \mathbb{R}^{d_{\text{model}} \times H d_v}$ to denote the combined Value projection across all heads in a layer.

2.3 Perturbation for Diverse Generation

The deterministic reasoning trajectory limits the diversity of solutions produced by repeated runs. One way to generate diverse trajectories is to inject random noise during inference. For example, You et al. (2025) add Gaussian noise directly to latent states to sample different reasoning paths. Gaussian perturbations can also be applied to model weights, as in parameter-space noise (Plappert et al., 2018). Activation noise directly changes the hidden states, whereas parameter noise changes the transformations applied to them. Both types of Gaussian perturbation can be written using the same operator. For an activation or weight matrix M, we define

$$\mathcal{P}_{\mathrm{G}}(M;\sigma) = M + \sigma\varepsilon, \qquad \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \tag{3}$$

where σ controls the perturbation scale. Section 3 builds on this idea by restricting parameter perturbations to the span of the rank-one components in a fixed SVD basis, rather than independently perturbing individual matrix entries.

3 SVP-V: Singular-Value Perturbation of Value Projections

Sampling distinct latent trajectories from an LRM requires stochasticity during latent reasoning. The challenge is to create meaningfully different trajectories without discarding the structure encoded by the model's learned transformations. We address this problem with **Singular-Value Perturbation (SVP)**. Instead of adding independent noise to individual weight entries, SVP randomly rescales the rank-one terms in the fixed SVD basis of the clean weight matrix while preserving its left and right singular vectors. Applying this perturbation to the attention Value projections described in Section 2.2 yields **SVP-V**. Section 3.1 first defines the perturbation and its trajectory-level sampling protocol. We then motivate the Value projection as the injection site in Section 3.2 and derive the multiplicative Gaussian distribution over SVD coefficients in Section 3.3.

3.1 Singular-Value Perturbation (SVP)

Let $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ have the singular value decomposition

$$W = U\Sigma R^\top, \qquad \Sigma = \mathrm{diag}(s), \tag{4}$$

where $r = \mathrm{rank}(W)$, $U \in \mathbb{R}^{d_{\text{in}} \times r}$ and $R \in \mathbb{R}^{d_{\text{out}} \times r}$ have orthonormal columns, and $s = [s_1, \ldots, s_r] \in \mathbb{R}^r$ contains the positive singular values. Let u_i and r_i denote the ith columns of U and R, respectively. Equivalently, $W = \sum_{i=1}^{r} s_i u_i r_i^\top$: each component maps input direction u_i to output direction r_i with coefficient s_i. The coefficient $s_i(1 + \alpha\varepsilon_i)$ can be negative. It is therefore a signed coefficient in the fixed SVD expansion of W, rather than necessarily a singular value of $\widetilde{W}$.

SVP samples $\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$ and rescales these coefficients while keeping the rank-one terms fixed:

$$\widetilde{W} = \sum_{i=1}^{r} s_i(1 + \alpha\varepsilon_i) u_i r_i^\top = W + \alpha \sum_{i=1}^{r} s_i \varepsilon_i u_i r_i^\top, \tag{5}$$

where $\alpha > 0$ controls the perturbation magnitude. Only the coefficients multiplying $u_i r_i^\top$ change; no cross terms $u_i r_k^\top$ for $i \neq k$ are introduced. Since the noise is zero-mean, $\mathbb{E}[\widetilde{W}] = W$.

SVP-V applies this construction to the combined Value projection W_V across all heads in every attention layer. For trajectory j and layer ℓ, we draw $\varepsilon_\ell^{(j)} \sim \mathcal{N}(0, I_{r_\ell})$ independently across layers and trajectories, where r_ℓ is the rank of that layer's Value projection. We hold these layer-wise

noise vectors fixed throughout prompt prefill and all latent reasoning steps. Thus, each trajectory evolves under one sampled set of perturbed Value projections. During answer decoding, newly generated tokens use the clean projections, while the attention cache retains the effects of the perturbed computation.

Although the analysis is presented for a single head, a perturbation of the combined Value projection can be partitioned into head-specific blocks, to which the same fixed-input argument applies. For inference efficiency, SVP-V can be implemented using cached right singular vectors without constructing dense matrices. Section A provides the factorized computation and pseudocode.

3.2 Why Perturb the Value Projection?

The routing–message distinction in Section 2.2 motivates perturbing the transmitted messages without directly changing the attention weights at the point of injection. Our choice of W_V combines this local structural argument with an empirical comparison of injection sites.

For a fixed input X, an additive perturbation to the Value projection produces the exact response

$$\widetilde{Y}(X) = P(X)X(W_V + \Delta W_V)W_O = Y(X) + P(X)X\Delta W_V W_O, \tag{6}$$

Thus, Value perturbations act on the OV circuit while leaving the attention matrix unchanged for the same input. For zero-mean perturbations, the local response is also zero-mean: $\mathbb{E}[\widetilde{Y}(X)] = Y(X)$. The same fixed-input argument applies to multi-head attention by partitioning the combined Value perturbation into head-specific blocks.

This routing preservation is local and conditional on a fixed input. The modified messages enter the residual stream and change the representations supplied to downstream Query and Key projections. They can therefore alter attention patterns in subsequent blocks and latent reasoning steps, even though the Query and Key parameters remain unchanged.

The local distinction is particularly relevant when attention is sharply concentrated. Let p_i denote the attention distribution for query i. Its softmax Jacobian, $\mathrm{diag}(p_i) - p_i p_i^\top$, approaches zero as p_i approaches a one-hot vector. Small Query- or Key-induced logit changes therefore have a diminished first-order effect on the attention weights. Value perturbations instead act directly on the transmitted messages without passing through this softmax Jacobian. Section B.1 provides quantitative bounds for this local comparison away from the one-hot limit.

These observations motivate perturbing the message side of attention, but do not establish a preference for W_V over W_O, since both belong to the OV circuit. We therefore make the remaining choice empirically: in the structure–location grid of Section C, W_V is best in 9 of 12 structure-controlled comparisons, including all six comparisons under SVP.

3.3 Why Spectrum-Scaled Gaussian Noise?

Having chosen the Value projection as the perturbation site, we now motivate the spectrum-scaled Gaussian noise used by SVP. We retain the learned SVD directions and consider additive perturbations to their coefficients:

$$\Delta W = U\,\mathrm{diag}(\eta)R^\top = \sum_{i=1}^{r} \eta_i u_i r_i^\top, \tag{7}$$

where η_i is the additive perturbation to the coefficient of $u_i r_i^\top$. Before choosing a distribution for these perturbations, we constrain their effect relative to the clean response for every input:

$$\mathbb{E}_\eta\left[\|x\Delta W\|_2^2\right] \le \alpha^2 \|xW\|_2^2, \qquad \text{for every } x \in \mathbb{R}^{d_{\text{in}}}. \tag{8}$$

We require the coefficient perturbations to have zero mean so that the perturbed weight remains centered at W. Under this condition, orthonormality of the singular vectors makes Equation (8) equivalent to $\mathrm{Var}(\eta_i) \le \alpha^2 s_i^2$ for every $i = 1, \ldots, r$. Thus, α controls the perturbation budget relative to each component's learned scale: components with small s_i, which contribute little to the clean response, receive proportionally little noise.

The response budget bounds these second moments, i.e., each coefficient's variance, but leaves the distributional shape and dependencies among coefficients unspecified. Suppose that η is absolutely

continuous with density q. Its differential entropy $\mathcal{H}(\eta) = -\int_{\mathbb{R}^r} q(z) \log q(z)\, dz$ measures how broadly probability mass is distributed in the fixed SVD-coefficient space. Following the Principle of Maximum Entropy (Jaynes, 1957), selecting the density that maximizes differential entropy ensures the perturbation introduces no arbitrary directional bias or unwarranted structural assumptions beyond the second-moment budget. Because multivariate Gaussians uniquely maximize differential entropy under covariance constraints (Cover & Thomas, 1991), the optimal coordinate distribution is necessarily zero-mean Gaussian.

Theorem 1 (Maximum-entropy SVD-coefficient distribution)**.** *Among all zero-mean, absolutely continuous distributions of η that induce perturbations of the form in Equation (7) and satisfy Equation (8), the unique distribution maximizing differential entropy is*

$$\eta_i = \alpha s_i \varepsilon_i, \qquad \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1). \tag{9}$$

The proof is provided in Section B.2. Since the perturbed coefficient is $s_i + \eta_i = s_i(1 + \alpha\varepsilon_i)$, Theorem 1 recovers exactly the multiplicative noise used by SVP. This distribution uses the full variance budget in every coordinate, so the response bound holds with equality for every input.

$$\mathbb{E}_\varepsilon \left[\|x\Delta W\|_2^2 \right] = \alpha^2 \|xW\|_2^2, \qquad \text{for every } x \in \mathbb{R}^{d_{\text{in}}}. \tag{10}$$

Thus, the RMS perturbation magnitude equals α times the clean response norm for every input.

Applied to each Value projection, Theorem 1 yields $\eta_\ell = \alpha s_\ell \odot \varepsilon_\ell$, with $\varepsilon_\ell \sim \mathcal{N}(0, I_{r_\ell})$. Entropy subadditivity further implies that independent layer-wise draws maximize the joint differential entropy under the per-layer response budgets. Thus, the fixed SVD basis determines the perturbation directions, the response budget determines their relative scale, and the maximum-entropy criterion determines the distribution.

4 Experiments

We evaluate SVP-V from three perspectives: test-time solution coverage, semantic variation in intermediate latent states, and reinforcement learning from perturbed latent trajectories.

Models and Benchmarks. COCONUT-GPT2 (Hao et al., 2024) is used in all three studies. The coverage study additionally evaluates CoDi-GPT2, CoDi-LLaMA-1B (Shen et al., 2025). The semantic analysis focuses on COCONUT-GPT2, while the reinforcement-learning study post-trains it and compares against GPT-2 variants of SLPO (You et al., 2026), SIM-CoT (Wei et al., 2025), and CoDi. Coverage and reinforcement learning use six arithmetic benchmarks: GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), MultiArith (Roy & Roth, 2015), SVAMP (Patel et al., 2021), ASDiv-A (Miao et al., 2020), and GSM-Plus (Li et al., 2024); the semantic analysis uses GSM8K, GSM-Hard, and SVAMP. For GSM-Plus, we use the test-mini subset excluding the critical-thinking category, whose questions lack a single numeric answer for exact-match grading.

4.1 Test-Time Solution Coverage

Experimental Setup. We compare SVP-V with deterministic greedy inference, answer-level Top-p sampling, additive Gaussian noise on activations and Value projections, and SVP on activations. For each model–method configuration, we select hyperparameters using mean pass@16 across five seeds on a GSM8K training subset of size 1,024 and then hold them fixed across all six evaluation benchmarks. Evaluation uses five unseen seeds and 64 samples per prompt to report the unbiased pass@16 estimator. Complete hyperparameter grids and selected values are provided in Section J.

Results. Table 1 reports pass@16 across three models and six benchmarks. SVP-V achieves the highest five-seed mean in 13 of 18 settings and outperforms Top-p sampling and activation Gaussian noise in all 18. It matches or exceeds Gaussian noise on the Value projections in 17 of 18 settings. Activation-space SVP leads in the remaining five settings, so an SVP-based method performs best in every setting. This trend extends to the five-model SIM-CoT suite, where SVP-V outperforms Top-p sampling in 29 of 30 settings (Section D). In the full structure–location grid, W_V is best in 9 of 12 model–benchmark–structure comparisons, including all six SVP comparisons, and SVP outperforms Gaussian noise in 34 of 42 comparisons with the perturbation location fixed (Section C).

Table 1: **Parallel test-time scaling performance (unbiased pass@16, %).** Reported as 5-seed mean ± std. The best numbers for each benchmark and model are in **bold**. See Section J for the selected hyperparameters and search grid. Evaluation on SIM-CoT is reported separately in Table 2.

Method	GSM8K	GSM-Hard	MultiArith	SVAMP	ASDiv-A	GSM-Plus
COCONUT-GPT2						
Deterministic	34.12	7.66	80.86	35.60	60.18	17.38
Top-p	53.76 ±0.24	12.44 ±0.08	88.53 ±0.43	49.97 ±0.33	72.58 ±0.37	38.95 ±0.34
Gaussian × Activation	54.25 ±0.16	12.69 ±0.13	93.41 ±0.12	50.45 ±0.28	72.57 ±0.28	36.43 ±0.21
SVP × Activation	54.90 ±0.18	13.10 ±0.13	93.34 ±0.30	52.86 ±0.40	73.81 ±0.14	37.56 ±0.16
Gaussian × W_V	55.98 ±0.25	13.59 ±0.28	93.46 ±0.29	51.68 ±0.41	74.13 ±0.18	39.16 ±0.30
SVP × W_V (SVP-V)	**57.22 ±0.48**	**13.90 ±0.12**	**93.60 ±0.07**	**53.45 ±0.38**	**74.87 ±0.30**	**40.05 ±0.34**
CoDi-GPT2						
Deterministic	42.46	9.26	91.90	40.00	65.44	23.10
Top-p	59.66 ±0.38	13.35 ±0.13	95.97 ±0.13	52.81 ±0.29	77.11 ±0.29	42.08 ±0.23
Gaussian × Activation	53.20 ±0.09	12.13 ±0.08	97.23 ±0.19	50.38 ±0.29	76.41 ±0.09	33.69 ±0.12
SVP × Activation	58.47 ±0.33	13.98 ±0.07	98.03 ±0.15	**57.73 ±0.31**	**80.47 ±0.13**	41.20 ±0.26
Gaussian × W_V	59.35 ±0.35	13.89 ±0.17	98.44 ±0.10	54.04 ±0.32	78.50 ±0.16	42.09 ±0.21
SVP × W_V (SVP-V)	**61.09 ±0.26**	**14.38 ±0.26**	**98.72 ±0.07**	56.14 ±0.31	79.99 ±0.35	**43.76 ±0.26**
CoDi-LLaMA-1B						
Deterministic	55.57	12.82	95.34	60.80	88.01	33.95
Top-p	68.01 ±0.23	16.35 ±0.14	97.32 ±0.08	69.86 ±0.25	93.31 ±0.09	49.96 ±0.23
Gaussian × Activation	65.30 ±0.10	15.43 ±0.14	97.79 ±0.04	68.24 ±0.23	93.38 ±0.05	44.98 ±0.11
SVP × Activation	69.51 ±0.21	**17.02 ±0.16**	98.62 ±0.08	**74.00 ±0.27**	**94.44 ±0.07**	50.92 ±0.18
Gaussian × W_V	70.14 ±0.28	16.68 ±0.08	**98.78 ±0.07**	72.86 ±0.14	94.43 ±0.10	51.96 ±0.12
SVP × W_V (SVP-V)	**70.29 ±0.35**	16.72 ±0.10	**98.78 ±0.05**	73.24 ±0.28	94.39 ±0.09	**52.25 ±0.13**

A controlled spectral-allocation ablation on COCONUT shows higher pass@16 when the learned coefficient scales are retained rather than flattened or permuted (Section I).

4.2 Semantic Branching

Inspecting COCONUT's latent states through the LM head reveals task-relevant intermediate quantities at certain reasoning steps. This observation motivates using LM-head readouts to examine semantic variation across sampled latent trajectories. For trajectory j at latent step t, we compute $p_t^{(j)} = \mathrm{softmax}(h_t^{(j)} W_{\mathrm{lm}}^{\top})$, where $h_t^{(j)}$ is the final-layer post-normalization state. We then quantify semantic diversity at each latent reasoning step as the average pairwise Jensen–Shannon divergence between the LM-head token distributions across 16 sampled trajectories.

Figure 2 (left) illustrates how this variation can yield successful alternatives when deterministic reasoning fails. The problem requires an intermediate total of 24 before reaching the final answer 6, but the clean trajectory reads out 22 at h_1 and ultimately answers 7. At the same step, SVP-V produces eight distinct top-1 readouts, compared with four under activation Gaussian noise. More importantly, the correct intermediate value 24 appears in six of 16 SVP-V trajectories, versus three under Gaussian noise, and four SVP-V trajectories ultimately reach the correct answer, while none of the Gaussian trajectories do. These observations suggest that SVP-V explores alternative intermediate computations that can recover from errors made by the clean trajectory. Additional examples are provided in Section G. Figure 2 (right) examines whether this increased semantic diversity holds more broadly across latent reasoning steps. SVP-V exhibits higher average pairwise JSD than activation Gaussian noise at every evaluated latent depth.

Together, these results suggest that intermediate LM-head readouts reveal semantic branching around decision-critical steps, where SVP-V explores alternative latent computations that can lead to different downstream outcomes. The readouts provide partial access to these intermediate computations without requiring each latent state to correspond to a discrete reasoning token.

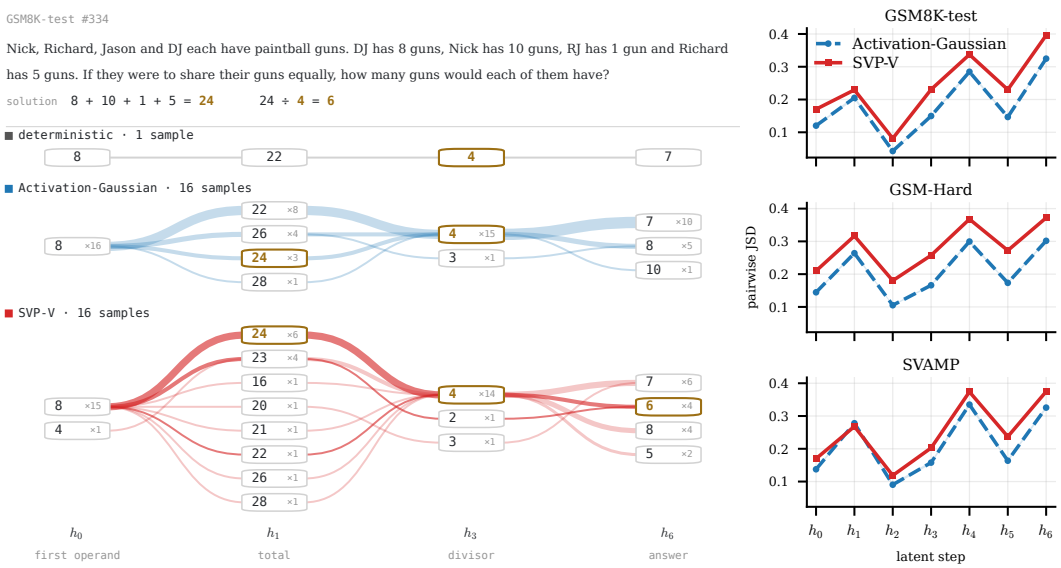

Figure 2: **Semantic branching in COCONUT. (Left)** SVP-V samples the correct intermediate value and develops several branches into the correct final answer. **(Right)** SVP-V produces greater pairwise JSD in LM-head readout distributions than activation Gaussian noise across benchmarks.

4.3 Reinforcement Learning with SVP-V Exploration

The preceding experiments show that SVP-V generates diverse latent trajectories containing task-relevant alternatives. We next examine whether these trajectories can provide a useful training signal. To this end, we incorporate SVP-V into Group Relative Policy Optimization (GRPO), yielding **SVP-V-GRPO**. The method uses outcome-based verifiable rewards to reinforce successful trajectories while updating only the Value projections, aligning the trainable parameters with the site of latent exploration.

Experimental Setup. We compare SVP-V-GRPO with three matched controls: Temp-GRPO, which obtains rollout diversity only from answer-token sampling; Gaussian-V-GRPO, which applies dense Gaussian perturbations to the Value projections; and Dropout-GRPO, which applies replayable MC-dropout during latent exploration (Jung, 2026). All four start from the same COCONUT-GPT2 checkpoint, update only the attention Value parameters, and use the same GSM8K-Aug dataset, group size, optimization settings, and reported checkpoint after nine epochs (Deng et al., 2024). Dropout-GRPO adapts prior work's exploration mechanism to this matched recipe; implementation details are given in Section F.2. For Gaussian-V-GRPO and SVP-V-GRPO, we transfer the perturbation scales selected to maximize unbiased pass@16 on the fixed GSM8K training subset (Sections 4.1 and J), without RL-specific scale tuning. We also compare with released SLPO, SIM-CoT, and CoDi checkpoints trained or post-trained on the same underlying training corpus (You et al., 2026; Wei et al., 2025; Shen et al., 2025).

Rollout Generation. For each rollout j, we independently draw the layer-wise noise vectors $\varepsilon_\ell^{(j)} \sim \mathcal{N}(0, I_{r_\ell})$ for $\ell = 1, \ldots, L$ and collect them as $\varepsilon^{(j)} = (\varepsilon_1^{(j)}, \ldots, \varepsilon_L^{(j)})$. We hold this noise fixed throughout prompt prefill and all recurrent latent steps. During answer decoding, we use the original unperturbed Value projections and sample answer tokens at temperature $T = 1$. For each prompt, we generate G rollouts and assign rollout j the binary reward $\mathcal{R}_j = \mathbb{1}\{o_j \text{ is correct}\}$. Following Dr.GRPO (Liu et al., 2025), we use the centered, unnormalized advantage $\widehat{A}_j = \mathcal{R}_j - \overline{\mathcal{R}}$, where $\overline{\mathcal{R}} = G^{-1} \sum_{k=1}^{G} \mathcal{R}_k$, and sum the policy-loss contributions over answer tokens without response-length normalization. Following DAPO's dynamic sampling (Yu et al., 2025), each iteration draws $2B$ candidate prompts and retains up to B groups containing both correct and incorrect answers. Under dynamic sampling, a group is retained only when it contains both correct and incorrect rollouts, since all-correct and all-incorrect groups have zero centered advantage. Group re-

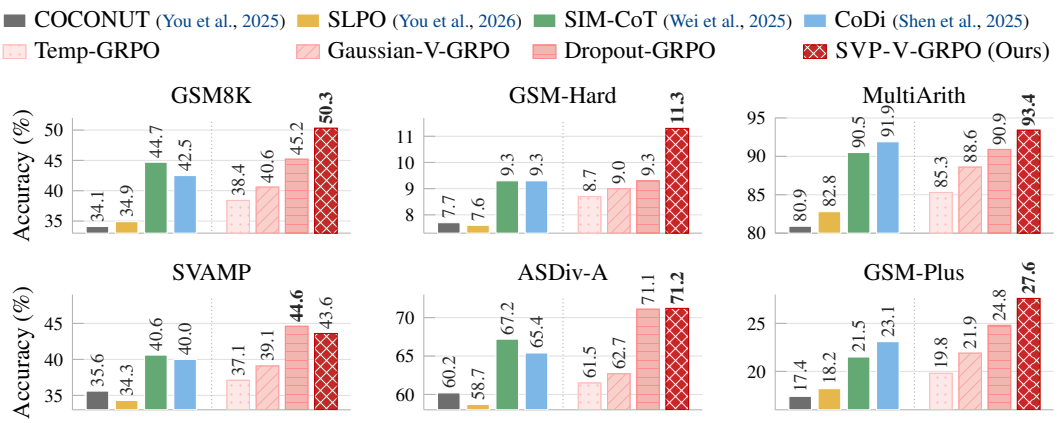

Figure 3: **Accuracy (%) of GPT-2-based LRMs and GRPO variants.** COCONUT, SLPO, SIM-CoT, and CoDi use checkpoints released by the cited works. Patterned bars denote GRPO variants initialized from the same COCONUT checkpoint and trained by us.

tention is therefore related to, but stricter than, pass@n: it measures how often exploration produces the reward contrast needed for learning. SVP-V increases this retention rate over answer-token sampling alone (Section F.2).

Policy Optimization. For each rollout, we store the noise $\varepsilon^{(j)}$ and sampled answer tokens. We then replay prompt prefill and latent reasoning with the same noise and recompute the answer-token log-probabilities from the stored prefixes. Gradients propagate through the replayed computation to the Value projections. Let $\pi_\theta(o \mid x, \varepsilon)$ denote the noise-conditioned answer policy. The current, rollout-time, and reference policies use the same rollout-specific noise and cached SVD factors. At answer position τ, the importance ratio is

$$\rho_{j,\tau} = \frac{\pi_\theta\left(o_{j,\tau} \mid x, \varepsilon^{(j)}, o_{j,<\tau}\right)}{\pi_{\theta_{\text{old}}}\left(o_{j,\tau} \mid x, \varepsilon^{(j)}, o_{j,<\tau}\right)}. \tag{11}$$

Because the noise distribution is independent of θ, its density cancels from the joint importance ratio when the same noise is used in the numerator and denominator. We optimize the clipped GRPO objective with KL regularization toward the initial reference policy under the same noise conditioning and periodically recompute the SVD factors as the parameters evolve. For GPT-2, the Value columns of the fused QKV matrices constitute approximately 5.7% of the model parameters. Complete details are provided in Section F.

Results. All reported accuracies use clean greedy decoding with perturbations disabled. Figure 3 compares SVP-V-GRPO with its starting COCONUT-GPT2 checkpoint, three released GPT-2-based continuous LRMs, and three matched GRPO controls. SVP-V-GRPO achieves best performance among all evaluated GPT-based continuous LRMs, and achieves highest accuracy on five of six benchmarks among the matched GRPO variants.

5 Conclusion

We introduced SVP, a structured perturbation method for exploring alternative latent trajectories, and instantiated it on attention Value projections as SVP-V. SVP-V expands solution coverage, while SVP-V-GRPO converts the resulting reward contrast into improvements that persist under unperturbed inference. These results establish structured latent exploration as an effective mechanism for both test-time sampling and RL post-training. More broadly, our results suggest that for latent models with little or no token-sampling diversity, structured parameter exploration can provide the behavioral variation needed for reinforcement learning. This opens a promising route toward effective RL post-training for a broader class of latent models that perform computation in continuous hidden states rather than through explicit token generation.

AI USE STATEMENT

We used generative AI tools to assist with implementing and editing research code, retrieving and identifying relevant literature, editing the manuscript for readability and clarity, and checking mathematical derivations and proofs for consistency and potential errors. Generative AI was used as an implementation and verification aid based on methods, theoretical arguments, and experimental procedures developed by the authors; it was not used to independently formulate research hypotheses, design experiments, generate synthetic data, formulate mathematical claims, construct proofs, evaluate experiments, or interpret results. All AI-assisted code was reviewed and tested by the authors, all mathematical derivations and proofs were independently verified by the authors, and all retrieved literature and AI-assisted text were independently checked. We take full responsibility for the final content of this work, including all text, claims, citations, code, proofs, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies exploration and reinforcement-learning post-training for continuous latent reasoning models using publicly available model checkpoints and arithmetic reasoning benchmarks. It does not involve human participants, personal or sensitive data, or deployment in high-stakes settings. We further support transparent verification by identifying the datasets and checkpoints used, distinguishing quoted results from re-evaluated results, and providing anonymized code, configurations, evaluation scripts, and the main trained checkpoint.

REPRODUCIBILITY STATEMENT

SVP and SVP-V are specified in detail in Section 3, and the inference procedure with cached right singular vectors is provided as pseudocode in Section A. The assumptions, proofs, and scope limitations of our theoretical results are provided in Section B. The evaluation protocol, including hyperparameter selection on the training subset, evaluation seeds, and the unbiased pass@n estimator, is described in Section 4.1. The hyperparameter search grids and selected values are listed in Section J, and the structure–location evaluation grid is provided in Section C. The SVP-V-GRPO objective, training recipe, training data, computational budget, and hardware configuration are described in Section F. Benchmark preprocessing is described in Section 4, and results quoted from prior work rather than re-evaluated are explicitly marked in the corresponding tables. Training and inference code, configurations, and evaluation scripts are available at https://anonymous.4open.science/r/Singular-Value-Perturbation-1234/, and the reported SVP-V-GRPO checkpoint is available at https://anonymous-hf.com/a/uoc7l9fxgpdc/.

REFERENCES

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 3

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168. 6

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991. 6

Jingcheng Deng, Zihao Wei, Liang Pang, Junhong Wu, Shicheng Xu, Zenghao Duan, and Huawei Shen. Latent-GRPO: Group relative policy optimization for latent reasoning. *CoRR*, abs/2604.27998, 2026. doi: 10.48550/arXiv.2604.27998. URL https://arxiv.org/abs/2604.27998. 3

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024. 8

Connor Dilgren and Sarah Wiegreffe. Are latent reasoning models easily interpretable? In *Conference on Language Modeling (COLM)*, 2026. 19, 20

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html. 3

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023. Introduces the GSM-Hard variant of GSM8K. 6

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC. 2

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *CoRR*, abs/2412.06769, 2024. doi: 10.48550/arXiv.2412.06769. URL https://arxiv.org/abs/2412.06769. 1, 3, 6

Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957. doi: 10.1103/PhysRev.106.620. 6

Wooil Jung. Dropout-GRPO: Variational stochasticity for continuous latent reasoning. *CoRR*, abs/2606.10184, 2026. doi: 10.48550/arXiv.2606.10184. URL https://arxiv.org/abs/2606.10184. 3, 8, 19

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yi-An Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. In *International Conference on Learning Representations*, 2026. 2

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.163. URL https://aclanthology.org/2024.acl-long.163/. 6

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *CoRR*, abs/2503.20783, 2025. doi: 10.48550/arXiv.2503.20783. URL https://arxiv.org/abs/2503.20783. 8

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/. 6

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/. 2

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 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*, pp. 2080–2094. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/. 6

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018. 4

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/. 6

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43276–43327. PMLR, 2024. URL https://proceedings.mlr.press/v235/sanford24a.html. 4

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 1

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. *CoRR*, abs/2502.21074, 2025. doi: 10.48550/arXiv.2502.21074. URL https://arxiv.org/abs/2502.21074. 1, 3, 6, 8, 9, 20

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. 2

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. *CoRR*, abs/2505.16552, 2025. doi: 10.48550/arXiv.2505.16552. URL https://arxiv.org/abs/2505.16552. 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 3

Jiecong Wang, Hao Peng, and Chunyang Liu. Latent chain-of-thought as planning: Decoupling reasoning from verbalization. *CoRR*, abs/2601.21358, 2026a. doi: 10.48550/arXiv.2601.21358. URL https://arxiv.org/abs/2601.21358. 2

Minghan Wang, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. Towards inference-time scaling for continuous space reasoning. *arXiv preprint arXiv:2510.12167*, 2025. 2

Minghan Wang, Ye Bai, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. GTS: Inference-time scaling of latent reasoning with a learnable gaussian thought sampler. *CoRR*, abs/2602.14077, 2026b. doi: 10.48550/arXiv.2602.14077. URL https://arxiv.org/abs/2602.14077. 2

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. *CoRR*, abs/2509.20317, 2025. doi: 10.48550/arXiv.2509.20317. URL https://arxiv.org/abs/2509.20317. 1, 3, 6, 8, 9, 17, 20

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. *CoRR*, abs/2510.07745, 2025. doi: 10.48550/arXiv.2510.07745. URL https://arxiv.org/abs/2510.07745. 2, 4, 9, 19

Runyang You, Zhiyuan Liu, Yongqi Li, and Wenjie Li. SLPO: Scaling latent reasoning via a surrogate policy. *CoRR*, abs/2607.19691, 2026. doi: 10.48550/arXiv.2607.19691. URL https://arxiv.org/abs/2607.19691. 6, 8, 9, 20

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/arXiv.2503.14476. URL https://arxiv.org/abs/2503.14476. 8

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. *CoRR*, abs/2403.09629, 2024. doi: 10.48550/arXiv.2403.09629. URL https://arxiv.org/abs/2403.09629. 2

Tunyu Zhang, Haizhou Shi, Yibin Wang, Hengyi Wang, Xiaoxiao He, Zhuowei Li, Haoxian Chen, Ligong Han, Kai Xu, Huan Zhang, Dimitris Metaxas, and Hao Wang. TokUR: Token-level uncertainty estimation for large language model reasoning. *arXiv preprint arXiv:2505.11737*, 2025. 2

Yuyan Zhou, Jiarui Yu, Hande Dong, Zhezheng Hao, Hong Wang, Jianqing Zhang, and Qiang Lin. LEPO: Latent reasoning policy optimization for large language models. *CoRR*, abs/2604.17892, 2026. doi: 10.48550/arXiv.2604.17892. URL https://arxiv.org/abs/2604.17892. 3

A Implementation Details of SVP-V

Algorithm 1 summarizes the Value-projection computation during SVP-V inference with cached right singular vectors. At model load time, we compute one SVD of every attention layer's Value projection, $W_{V,\ell} = U_\ell \operatorname{diag}(s_\ell) R_\ell^\top$, cache only R_ℓ, and discard U_ℓ and s_ℓ. This is sufficient because $W_{V,\ell} R_\ell = U_\ell \operatorname{diag}(s_\ell)$. Each trajectory j independently samples $\varepsilon_\ell^{(j)} \sim \mathcal{N}(0, I_{r_\ell})$ for every layer ℓ and keeps the resulting layer-wise collection fixed from prompt prefill through the final latent step. For a row input x at layer ℓ, let $y = xW_{V,\ell}$ denote the clean Value-projection output. The perturbation can then be computed exactly as

$$x\widetilde{W}_{V,\ell}^{(j)} = y + \alpha \left((yR_\ell) \odot \varepsilon_\ell^{(j)} \right) R_\ell^\top . \tag{12}$$

The perturbed dense matrix is never materialized, and the clean projection y is reused in the residual computation. We apply Equation (12) during prompt prefill and latent reasoning. During answer decoding, newly generated tokens use the clean output y, while the attention cache retains the effects of the perturbed computation. This R_ℓ-only cache is exact for inference because the Value projections remain fixed after their one-time factorization.

Measured inference overhead. On COCONUT-GPT2, the R-only implementation requires a 27.0 MiB persistent factor cache, compared with 54.0 MiB when U, s, and R are all retained, reducing factor-cache memory by 50.0%. Under a fixed fp32 GSM8K-test workload with 16 trajectories per problem and batch size 8, its synchronized generation latency is $27.5 \pm 1.1\%$ higher than clean inference. This value is the mean and standard deviation over three repetitions on each of two NVIDIA RTX PRO 6000 Blackwell GPUs after warm-up. The one-time SVD preprocessing cost is excluded because it is incurred when the model is loaded rather than during generation.

B Mathematical Theory behind SVP-V

This section develops the routing–message comparison in Section 3.2 and proves the maximum-entropy characterization in Section 3.3. The claims are conditional: they concern local responses in the fixed parameterization of a trained model and the specified spectral perturbation class, rather than a universal ranking over all parameterizations or exploration distributions.

Algorithm 1 SVP-V inference using cached right singular vectors

Require: Value projections $\{W_{V,\ell}\}_{\ell=1}^{L}$, perturbation scale α, number of trajectories n
Ensure: Value-projection outputs for n independently perturbed trajectories

for $\ell = 1, \ldots, L$ **do** ▷ one-time preprocessing
  $(U_\ell, s_\ell, R_\ell^\top) \leftarrow \mathrm{SVD}(W_{V,\ell})$
  Cache R_ℓ; discard U_ℓ and s_ℓ
for $j = 1, \ldots, n$ **do**
  for $\ell = 1, \ldots, L$ **do**
    Sample $\varepsilon_\ell^{(j)} \sim \mathcal{N}(0, I_{r_\ell})$ ▷ fixed throughout trajectory j
  for each inference step t and attention layer ℓ **do**
    $x \leftarrow x_{t,\ell}$
    $y \leftarrow xW_{V,\ell}$ ▷ clean Value-projection output
    if t belongs to prefill or latent reasoning **then**
      $y_{t,\ell} \leftarrow y + \alpha\Big((yR_\ell) \odot \varepsilon_\ell^{(j)}\Big)R_\ell^\top$
    else ▷ explicit answer decoding
      $y_{t,\ell} \leftarrow y$

B.1 Finite-Concentration QK–OV Circuit Comparison

Following the QK/OV circuit distinction in Section 2.2, we compare how perturbations to attention routing (QK) and message transformation (OV) affect the output of a single attention head. Throughout this comparison, the block input X is fixed.

Fix a query position i, and let source indices range over the K_i positions visible to it. Let $z_i \in \mathbb{R}^{K_i}$ be the column vector of pre-softmax attention logits for query i in Equation (2), and let $p_i = \mathrm{softmax}(z_i)$ be the corresponding column vector of attention weights. Define $H_i := \mathrm{diag}(p_i) - p_i p_i^\top$ as the softmax Jacobian, so that a QK-induced logit perturbation δz_i changes p_i by $H_i \delta z_i$ to first order. As p_i approaches a one-hot vector, $H_i \to 0$, which is the formal content of the observation in Section 3.2.

Let $x_j^\top$ be row j of X, and let $B := W_V W_O$ denote the OV transformation. The message from source position j is $m_j := B^\top x_j$, and the attention-head output at query position i is $y_i := \sum_j p_{ij} m_j$, so that $y_i^\top$ is row i of $Y(X)$. An OV perturbation ΔB changes each message by $\Delta m_j := \Delta B^\top x_j$ while leaving p_i unchanged. For SVP-V, $\Delta B = \Delta W_V W_O$.

Let Δy_i^{OV} denote the exact change in the attention-head output caused by ΔB with the attention weights fixed. Let δy_i^{QK} denote the first-order change in the same output caused by δz_i with the messages fixed. Here, Δ denotes an exact finite change and δ denotes a first-order response. These responses are given by

$$\Delta y_i^{\mathrm{OV}} = \sum_j p_{ij} \Delta m_j, \qquad \delta y_i^{\mathrm{QK}} = \sum_j (H_i \delta z_i)_j \, m_j = \sum_j p_{ij} \delta z_{ij} (m_j - y_i), \tag{13}$$

Assume $K_i \geq 2$; when $K_i = 1$, the QK response is exactly zero and the OV response equals the message change at the sole visible position. Let $j^\star \in \arg\max_j p_{ij}$, so that $1 - p_{ij^\star}$ is the attention mass off the top position, and let $\gamma_i := z_{ij^\star} - \max_{j \neq j^\star} z_{ij}$ be the logit margin, $D_i := \max_{j,k} \|m_j - m_k\|_2$, and $L_i := \max_j \|\Delta m_j\|_2$. Separating the contribution of $j^\star$ from the remaining positions and using the triangle inequality gives the first two bounds below. The third follows from the softmax probability ratios and the logit margin.

$$\begin{aligned} \|\delta y_i^{\mathrm{QK}}\|_2 &\leq 2(1 - p_{ij^\star})\, D_i \|\delta z_i\|_\infty, \\ \|\Delta y_i^{\mathrm{OV}} - \Delta m_{j^\star}\|_2 &\leq 2(1 - p_{ij^\star})\, L_i, \\ 1 - p_{ij^\star} &\leq (K_i - 1) e^{-\gamma_i}. \end{aligned} \tag{14}$$

For bounded D_i, $\|\delta z_i\|_\infty$, and L_i, these bounds show that the first-order QK response vanishes as attention concentrates, while the exact OV response approaches $\Delta m_{j^\star}$. Thus, concentrated attention suppresses the local effect of QK-induced logit changes, whereas an OV perturbation directly

changes the message transmitted from the dominant source position. This is a local comparison of the QK and OV computations, not a parameterization-independent ranking of W_Q, W_K, W_V, and W_O; accordingly, the choice of W_V over W_O within the OV circuit is evaluated empirically in Section C.

B.2 Proof of the Maximum-Entropy Characterization

Proof of Theorem 1. For the perturbation in Equation (7), orthonormality of the right singular vectors gives

$$\begin{aligned} \mathbb{E}\big[\|x\Delta W\|_2^2\big] &= \sum_{i=1}^{r} \mathbb{E}[\eta_i^2](xu_i)^2, \\ \|xW\|_2^2 &= \sum_{i=1}^{r} s_i^2 (xu_i)^2. \end{aligned} \tag{15}$$

Because η is zero-mean, taking $x = u_i^\top$ shows that the uniform response budget implies $\mathrm{Var}(\eta_i) \le \alpha^2 s_i^2$ for every i. Conversely, these coordinate-wise bounds imply the response budget through the expansion above.

Let $C = \mathrm{Cov}(\eta)$. Then

$$\begin{aligned} \mathcal{H}(\eta) &\overset{(a)}{\le} \frac{1}{2}\log((2\pi e)^r \det C) \\ &\overset{(b)}{\le} \frac{1}{2}\log\left((2\pi e)^r \prod_{i=1}^{r} C_{ii}\right) \\ &\overset{(c)}{\le} \frac{1}{2}\log\left((2\pi e)^r \alpha^{2r} \prod_{i=1}^{r} s_i^2\right). \end{aligned} \tag{16}$$

Here, (a) follows from the Gaussian maximum-entropy theorem for a fixed covariance, (b) from Hadamard's determinant inequality $\det C \le \prod_i C_{ii}$, and (c) from the coordinatewise variance constraints $C_{ii} \le \alpha^2 s_i^2$. Equality throughout holds only for a Gaussian with $C = \alpha^2 \,\mathrm{diag}(s_1^2, \dots, s_r^2)$. Thus the unique maximizer is $\eta_i = \alpha s_i \varepsilon_i$ with independent $\varepsilon_i \sim \mathcal{N}(0,1)$, as stated in Equation (9). □

Let η_ℓ denote the coefficient-perturbation vector at layer ℓ, and let $\eta_{\mathrm{all}} := (\eta_\ell)_{\ell=1}^{L}$ denote their concatenation, as in Section 3.3. For any absolutely continuous joint distribution with zero-mean layer-wise vectors satisfying the per-layer response budgets, entropy subadditivity gives

$$\mathcal{H}(\eta_{\mathrm{all}}) \le \sum_{\ell=1}^{L} \mathcal{H}(\eta_\ell), \tag{17}$$

with equality if and only if the layer-wise coefficient vectors are independent. Since each marginal is uniquely maximized by Equation (9), their independent product is the unique joint maximum-entropy law, matching the layer-wise sampling rule used by SVP-V.

B.3 Limitations of the Local Analysis

First, the local routing–message comparison in Section B.1 does not establish a theoretical preference for W_V over W_O; that choice is evaluated empirically. Second, our implementation leaves $1 + \alpha\varepsilon_i$ unclipped, so a sufficiently negative draw can reverse the sign of a coefficient in the fixed SVD expansion; the corresponding rank-one outer product $u_i r_i^\top$ is still retained, but such a draw need not be a small, nonnegative singular-value change. Third, maximum entropy in spectral-coefficient space is not the same as maximum task utility or semantic entropy. A learned, prompt-dependent covariance could allocate noise toward more useful downstream directions. Finally, the routing–message analysis holds the block input fixed. It does not characterize how perturbations propagate through subsequent layers or guarantee changes in the resulting reasoning trajectory.

C Perturbation Structure and Location

The main evaluation compares a small set of representative methods, which changes perturbation structure and injection location together. We therefore run a complete structure–location grid to separate these two choices.

C.1 Experimental Protocol

We run the full grid on COCONUT and CoDi-GPT2 using GSM8K, GSM-Hard, and MultiArith. We compare additive Gaussian noise and SVP at seven injection sites, comprising the recurrent latent activation, W_Q, W_K, W_V, W_O, W_{up}, and W_{down}. At weight sites, each perturbation acts on the target matrix.

At the activation site, we compare activation Gaussian noise with a data-dependent variant of SVP. This variant computes the SVD of the current batch of hidden states and multiplicatively perturbs each state's coordinates in the resulting right-singular basis by $1 + \alpha\varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(0, 1)$. Because the basis depends on the sampled batch rather than fixed learned weights, trajectories within a batch share a data-dependent basis and their perturbations are not mutually independent.

Every structure–location cell receives its own scale selection. We maximize five-seed mean pass@16 on the fixed 1,024-problem GSM8K training subset using seeds 0–4, freeze the selected value, and evaluate with the disjoint seeds 5–9. As in Section 4.1, each problem produces $N = 64$ candidates and performance is measured with the unbiased pass@16 estimator. The selected hyperparameters are listed in Table 6.

C.2 Results Across Structures and Locations

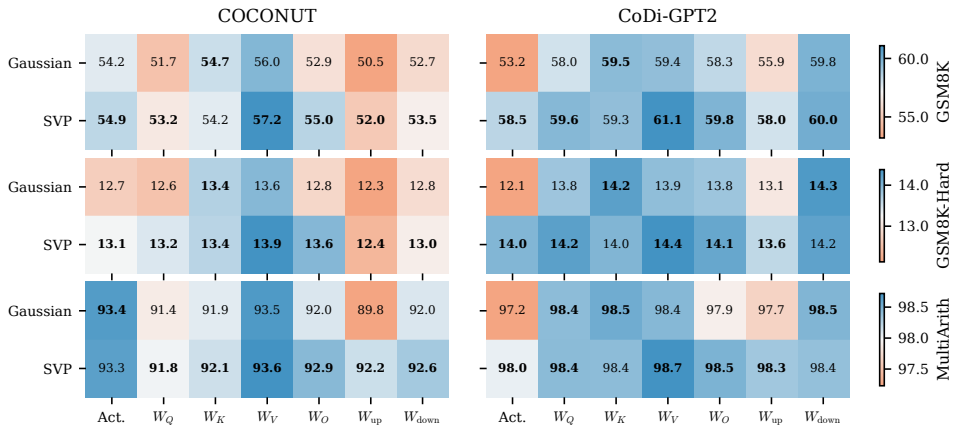

Figure 4: **Pass@16 (%) across perturbation structures and injection sites.** Bold indicates the best-performing structure per injection site for each model and benchmark.

Figure 4 reports all 84 evaluated cells (2 models $\times$ 3 benchmarks $\times$ 2 structures $\times$ 7 locations). Holding the perturbation structure fixed, W_V achieves the highest mean in 9 of the 12 model–benchmark–structure comparisons, including all six SVP comparisons. In the three CoDi-GPT2 Gaussian comparisons, W_K or W_{down} leads W_V by at most 0.46 percentage points. Holding the location fixed, SVP outperforms Gaussian noise in 34 of the 42 model–benchmark–location comparisons. Both the effectiveness of the Value projection and the overall advantage of SVP persist on GSM-Hard and MultiArith without retuning the perturbation scales.

C.3 Interpretation

The two axes support different conclusions. The location comparison shows that the Value projection is a consistently effective intervention point, including for non-spectral noise. At weight sites, SVP generally outperforms entry-wise Gaussian noise, supporting the combined use of fixed

Table 2: **Parallel test-time scaling on the SIM-CoT model suite (unbiased pass@16, %).** The protocol follows Section 4.1; results are 5-seed means ± standard deviations.

Method	GSM8K	GSM-Hard	MultiArith	SVAMP	ASDiv-A	GSM-Plus
SIM-CoT-GPT2-COCONUT						
Deterministic	44.66	9.33	90.52	40.60	67.24	21.52
Top-p	54.97 ±0.33	12.24 ±0.12	94.50 ±0.12	49.99 ±0.21	74.86 ±0.12	34.96 ±0.28
Gaussian × Activation	58.26 ±0.15	13.25 ±0.10	96.08 ±0.14	51.53 ±0.11	75.21 ±0.07	35.87 ±0.19
SVP × Activation	58.80 ±0.19	13.38 ±0.10	96.67 ±0.14	54.57 ±0.33	77.97 ±0.32	37.03 ±0.21
SVP × W_V (SVP-V)	**59.90 ±0.21**	**13.54 ±0.12**	**97.37 ±0.08**	**54.78 ±0.35**	**78.51 ±0.19**	**38.34 ±0.18**
SIM-CoT-GPT2-CoDi						
Deterministic	39.35	8.73	91.03	40.10	64.29	22.57
Top-p	57.99 ±0.15	13.11 ±0.04	94.15 ±0.20	52.91 ±0.36	76.48 ±0.13	41.47 ±0.19
Gaussian × Activation	54.18 ±0.32	12.47 ±0.14	94.57 ±0.12	51.30 ±0.39	76.63 ±0.20	36.23 ±0.21
SVP × Activation	56.54 ±0.31	13.03 ±0.13	95.86 ±0.40	**56.28 ±0.32**	78.73 ±0.12	39.40 ±0.21
SVP × W_V (SVP-V)	**60.26 ±0.30**	**14.08 ±0.17**	**96.60 ±0.17**	56.14 ±0.28	**79.32 ±0.14**	**42.34 ±0.12**
SIM-CoT-LLaMA-1B						
Deterministic	55.88	12.52	96.03	61.00	86.70	32.90
Top-p	66.58 ±0.24	15.34 ±0.08	97.47 ±0.03	70.96 ±0.15	91.16 ±0.11	47.97 ±0.16
Gaussian × Activation	66.62 ±0.22	15.44 ±0.12	98.44 ±0.14	69.78 ±0.23	91.68 ±0.09	45.61 ±0.10
SVP × Activation	**70.57 ±0.46**	**16.57 ±0.19**	**99.54 ±0.06**	**75.32 ±0.22**	**94.16 ±0.14**	**51.12 ±0.22**
SVP × W_V (SVP-V)	68.88 ±0.16	16.06 ±0.12	98.75 ±0.08	74.83 ±0.19	93.40 ±0.08	49.73 ±0.29
SIM-CoT-LLaMA-3B						
Deterministic	61.79	14.57	98.79	74.50	93.19	39.52
Top-p	71.42 ±0.15	17.95 ±0.11	99.00 ±0.05	80.00 ±0.19	95.53 ±0.04	52.94 ±0.16
Gaussian × Activation	73.00 ±0.15	17.77 ±0.07	99.22 ±0.01	82.66 ±0.07	95.77 ±0.05	53.92 ±0.14
SVP × Activation	74.36 ±0.17	18.70 ±0.29	**99.45 ±0.07**	83.12 ±0.12	96.07 ±0.07	55.60 ±0.15
SVP × W_V (SVP-V)	**75.33 ±0.17**	**18.73 ±0.11**	99.35 ±0.04	**84.91 ±0.10**	**96.36 ±0.06**	**57.81 ±0.11**
SIM-CoT-LLaMA-8B						
Deterministic	62.40	15.93	99.83	79.30	94.99	42.95
Top-p	75.76 ±0.20	20.95 ±0.13	100.00 ±0.00	83.88 ±0.25	96.69 ±0.09	58.01 ±0.24
Gaussian × Activation	76.93 ±0.21	21.05 ±0.17	100.00 ±0.00	88.08 ±0.13	97.82 ±0.03	58.67 ±0.21
SVP × Activation	77.32 ±0.24	21.38 ±0.07	100.00 ±0.00	**88.96 ±0.22**	**97.87 ±0.08**	59.85 ±0.14
SVP × W_V (SVP-V)	**78.40 ±0.18**	**21.64 ±0.06**	100.00 ±0.00	88.83 ±0.16	97.50 ±0.04	**61.14 ±0.34**

SVD directions and spectrum-scaled perturbations. The activation column separately evaluates the data-dependent variant defined above.

Within the W_V column, SVP exceeds entry-wise Gaussian noise by 1.2, 0.3, and 0.1 percentage points on COCONUT (GSM8K, GSM-Hard, MultiArith) and by 1.7, 0.5, and 0.3 points on CoDi-GPT2. The gains from switching the injection site from activations to W_V are generally larger than those from replacing Gaussian noise with SVP at W_V. These results empirically support the two design choices of SVP-V without establishing formal optimality.

D Solution Coverage on SIM-CoT Models

We evaluate five checkpoints from the official SIM-CoT release (Wei et al., 2025), comprising two GPT-2 variants (SIM-CoT-GPT2-COCONUT and SIM-CoT-GPT2-CoDi) and three CoDi-based SIM-CoT-LLaMA variants at 1B, 3B, and 8B.[1] At inference, the auxiliary training decoder is removed, and each SIM-CoT model performs sequential latent reasoning. We therefore use the same inference procedure, hyperparameter-selection protocol, and evaluation seeds as in Section 4.1. Table 2 reports pass@16 for all five released checkpoints.

Across the 30 model–benchmark settings, SVP-V exceeds Top-p sampling in 29 and ties it at 100% on MultiArith with SIM-CoT-LLaMA-8B. Activation-space SVP remains competitive and achieves higher mean pass@16 than SVP-V on all six benchmarks for SIM-CoT-LLaMA-1B.

[1] https://huggingface.co/collections/Wiselnn/sim-cot-supervised-implicit-chain-of-thought

E PASS@n SCALING CURVES

Table 1 reports pass@16 only. Figure 5 plots pass@n for $n \in \{2, 4, 8, 16, 32\}$ on GSM8K-test for COCONUT and CoDi-LLaMA-1B, using the four Table 1 rows Deterministic, Top-p, Gaussian $\times$ Activation, and SVP-V. The curves are computed from the same runs as Table 1 (unbiased estimator over $N = 64$ candidates, train-selected hyperparameters, evaluation seeds 5–9).

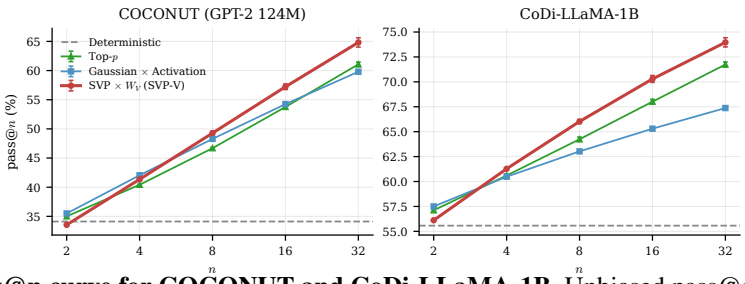

Figure 5: **Pass@n curve for COCONUT and CoDi-LLaMA-1B.** Unbiased pass@n over $N = 64$ candidates per problem at the train-selected hyperparameters in Table 1, 5-seed mean $\pm$ std. The dashed line is deterministic greedy accuracy, which is independent of n.

At $n = 2$, the stochastic methods remain close to deterministic decoding, and SVP-V has the lowest mean among the stochastic methods. It is 0.6 percentage points below deterministic decoding on COCONUT and 0.5 points above it on CoDi-LLaMA-1B, while Top-p and activation Gaussian noise are 0.9–1.9 points above. The perturbation scales are selected for pass@16, so they need not be optimal at smaller sampling budgets.

SVP-V shows the largest overall increase in pass@n from $n = 2$ to $n = 32$ on both models. On COCONUT, it overtakes activation Gaussian noise at $n = 8$ (49.3% vs. 48.3%) and exceeds it by 3.0 and 5.0 percentage points at $n = 16$ and $n = 32$, respectively. On CoDi-LLaMA-1B, its margin over Top-p is 0.7 points at $n = 4$ and 2.2 points at $n = 32$. Activation Gaussian noise shows smaller gains at larger n on this model, reaching 67.4% at $n = 32$, compared with 71.7% for Top-p and 74.0% for SVP-V. At the selected hyperparameters, SVP-V achieves the highest mean coverage among the compared methods from $n = 8$ on COCONUT and $n = 4$ on CoDi-LLaMA-1B across the evaluated sampling budgets.

F IMPLEMENTATION DETAILS FOR SVP-V-GRPO

This section complements Section 4.3 with the full optimization objective, training configuration, matched GPT-2 controls, and additional LLaMA results.

F.1 FULL TRAINING OBJECTIVE

For each retained rollout, we reuse its stored noise collection $\varepsilon^{(j)}$ when evaluating the current policy π_θ, rollout-time policy $\pi_{\theta_{\text{old}}}$, and reference policy π_{θ_0}. The SVD factors used to generate the rollout remain fixed during replay, so all three policies receive the same additive perturbation. The reference parameters θ_0 remain fixed at the initial SFT checkpoint. Using the ratio $\rho_{j,\tau}$ from Equation (11), the loss for one prompt group is

$$\mathcal{L}_{\text{SVP-GRPO}} = -\frac{1}{G}\sum_{j=1}^{G}\sum_{\tau\in\mathcal{T}_j}\min\Big(\rho_{j,\tau}\widehat{A}_j, \text{clip}(\rho_{j,\tau}, 1-\epsilon, 1+\epsilon)\widehat{A}_j\Big) + \beta\frac{\sum_{j=1}^{G}\sum_{\tau\in\mathcal{T}_j}\widehat{D}^{(k_3)}_{\text{KL},j,\tau}}{\sum_{j=1}^{G}|\mathcal{T}_j|}, \tag{18}$$

where $\mathcal{T}_j$ contains the generated answer tokens for rollout j, including EOS when present and excluding padding. The policy term sums over these tokens and divides by G; it does not normalize individual responses by length. The KL term is instead averaged over all valid answer tokens. We

then average the loss over retained prompt groups. Let

$$\delta_{j,\tau} = \log \pi_{\theta_0}\Big(o_{j,\tau} \mid w, \varepsilon^{(j)}, o_{j,<\tau}\Big) - \log \pi_{\theta}\Big(o_{j,\tau} \mid w, \varepsilon^{(j)}, o_{j,<\tau}\Big). \tag{19}$$

The nonnegative k_3 estimator is

$$\widehat{D}^{(k_3)}_{\mathrm{KL},j,\tau} = \exp(\delta_{j,\tau}) - \delta_{j,\tau} - 1. \tag{20}$$

We clamp $\delta_{j,\tau}$ to $[-10, 10]$ before evaluating Equation (20). At each refresh, we compute the SVD factors from the current policy and use them with the stored noise $\varepsilon^{(j)}$ to construct the same additive perturbation for the current, rollout-time, and reference policies. The reference base weights θ_0 remain fixed at the initial SFT checkpoint. This common-random-numbers construction ensures that the ratio and KL term do not also reflect variation from independent perturbation draws.

F.2 Training and Optimization Details

Initialization checkpoints. The GPT-2 experiments start from the public COCONUT checkpoint released with Parallel Test-Time Scaling (You et al., 2025).[2] The LLaMA-3.2-1B experiment starts from the public COCONUT checkpoint released by Dilgren & Wiegreffe (2026).[3]

Common training configuration. The main SVP-V-GRPO runs and matched GPT-2 controls use 383,891 eligible GSM8K-Aug prompts, six latent steps, and $G = 32$ rollouts per prompt. Each iteration samples $2B$ candidate prompts and retains up to B groups containing both correct and incorrect answers. Answer tokens are sampled with temperature $T = 1$ and top-$p = 1$. Each retained buffer is reused for $\mu = 2$ optimization passes.

We use centered advantages without additional reweighting, symmetric clipping with $\epsilon = 0.2$, and a k_3 KL penalty with coefficient $\beta = 0.02$ and log-difference clamp $[-10, 10]$. Optimization uses AdamW with a constant learning rate, no warmup or weight decay, and gradient clipping at 1.0. Only the attention Value projections are trainable; all other parameters remain frozen.

The GPT-2 runs use $B = 32$, a learning rate of 6×10^{-5}, and seed 0. The LLaMA-3.2-1B run uses $B = 64$ and a learning rate of 2×10^{-5}, with results reported at epoch seven. Both SVP-V-GRPO configurations use $\alpha = 0.6$. Their SVD factors are recomputed every 50 iterations before rollout collection and remain fixed throughout optimization over that buffer. Perturbations are added to the current Value weights and remain zero-mean between refreshes.

Matched GPT-2 controls. We compare four exploration mechanisms under the common configuration above. Temp-GRPO obtains diversity solely through answer-token sampling. Gaussian-V-GRPO applies dense Gaussian perturbations to the Value projections with $\sigma = 0.06$, while SVP-V-GRPO uses the spectrum-scaled perturbations in Equation (5). For GPT-2, both scales are the values selected to maximize five-seed mean pass@16 on the fixed 1,024-problem GSM8K training subset (Sections 4.1 and J), transferred to GRPO without RL-specific scale tuning.

Dropout-GRPO adapts the exploration mechanism of Jung (2026) to this recipe. It applies MC-Dropout at the attention residual-output and MLP-output sites in all 12 layers, using $p = 0.2$ and inverted-dropout scaling $1/(1-p)$. Each rollout stores a seed for regenerating its masks during policy replay. Prefill masks are indexed by token position, while each site's mask is held fixed across recurrent latent steps. Dropout is disabled during answer decoding while the perturbed context cache is retained.

All four variants are evaluated using clean greedy decoding with exploration disabled. Figure 6 shows their per-epoch accuracies on all six benchmarks.

Training budget and retained groups. We report all four GPT-2 variants at epoch nine. Each iteration uses the same budget of 64 candidate prompts and 2,048 reward evaluations. Temp-GRPO, Gaussian-V-GRPO, Dropout-GRPO, and SVP-V-GRPO retain an average of 13.1, 25.1, 26.9, and 31.8 groups per iteration, respectively. The loss is averaged over retained groups (Section F.1),

[2] https://huggingface.co/ModalityDance/latent-tts-coconut
[3] https://huggingface.co/connordilgren/llama32-1b-gsm8k-coconut

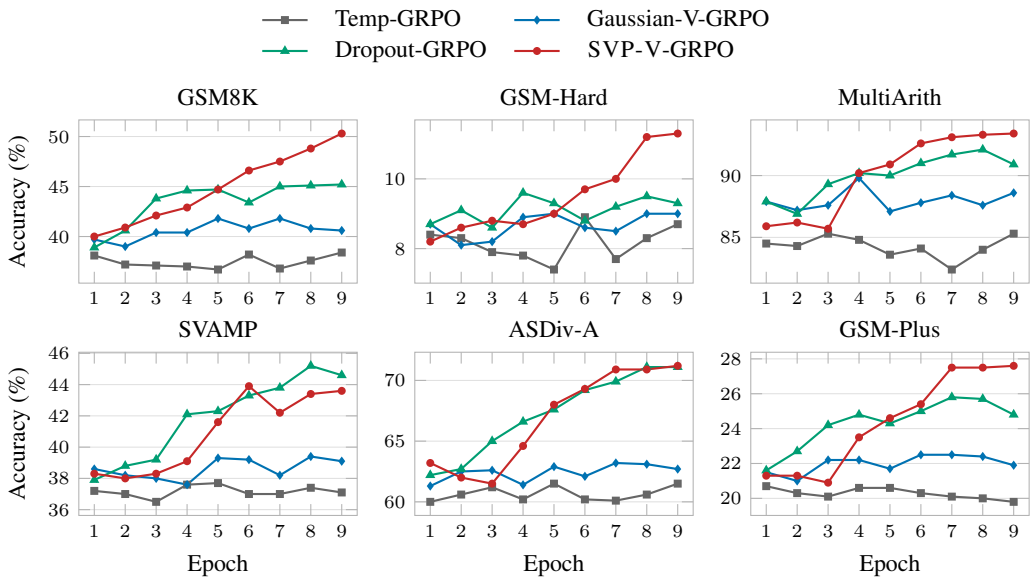

Figure 6: **Learning curves of the GRPO exploration ablation.** Accuracy (%) after each epoch for the four GRPO variants of Figure 3. All evaluations use clean greedy decoding with perturbations disabled; epoch 9 is the reported checkpoint.

so these differences affect the effective batch size while the optimizer-step budget remains fixed. Because SVP-V-GRPO frequently reaches the cap of 32, its retained count is a lower bound on the number of mixed-outcome groups generated. Retention therefore measures how often exploration produces a usable learning signal under the group filter.

The corresponding six-benchmark mean accuracies are 41.8%, 43.7%, 47.6%, and 49.6%. Gaussian-V-GRPO retains 1.9× as many groups as Temp-GRPO and improves mean accuracy by 1.9 percentage points, whereas SVP-V-GRPO retains 1.27× as many groups as Gaussian-V-GRPO and improves it by a further 5.9 points. These differences suggest that retained-group count alone provides an incomplete account of the results in Figure 3.

F.3 Results on COCONUT-LLaMA-3.2-1B

Under our compute budget, we evaluate the single LLaMA-3.2-1B configuration described in Section F.2. Table 3 reports clean greedy accuracy for the released COCONUT checkpoint, SVP-V-GRPO, and the compared continuous LRMs. The SLPO and SIM-CoT entries are quoted from their papers and cover only the reported benchmarks; the remaining entries are evaluated in our harness.

Table 3: Accuracy (%) of SVP-V-GRPO and continuous LRMs with a LLaMA-3.2-1B backbone under their native latent inference (clean greedy decoding). Released checkpoints are evaluated in our harness; SVP-V-GRPO starts from the released COCONUT-LLaMA-3.2-1B checkpoint and reports the single configuration evaluated under our compute budget. Results marked † and ‡ are quoted from You et al. (2026) and Wei et al. (2025), respectively; these entries were not re-evaluated in our harness, and unreported entries are marked "–".

	GSM8K	GSM-Hard	MultiArith	SVAMP	ASDiv-A	GSM-Plus
COCONUT (base checkpoint) (Dilgren & Wiegreffe, 2026)	35.9	8.2	84.1	48.5	77.8	20.6
+ SLPO† (You et al., 2026)	22.4	5.8	46.9	–	–	–
+ SIM-CoT‡ (Wei et al., 2025)	–	9.3	87.7	43.9	–	–
+ Self-Distillation (CoDi) (Shen et al., 2025)	55.6	**12.8**	95.3	**60.8**	**88.0**	34.0
+ SVP-V-GRPO	**56.6**	**12.8**	**97.8**	58.0	87.1	**34.8**

SVP-V-GRPO improves GSM8K accuracy from 35.9% to 56.6% and exceeds the base checkpoint on all six benchmarks. Among the compared LRMs, it achieves the highest accuracy on GSM8K,

MultiArith, and GSM-Plus and ties CoDi on GSM-Hard. CoDi remains higher on SVAMP and ASDiv-A.

F.4 Value-Only Updates versus Attention and MLP LoRA

Table 4: Parameter update scope comparison on COCONUT-GPT-2 after nine epochs. Both runs use SVP-V exploration; the LoRA run adapts attention and MLP projections while freezing base weights. Mean is the unweighted mean over the six benchmarks.

Update scope	LR	GSM8K	GSM-Hard	MultiArith	SVAMP	ASDiv-A	GSM-Plus	Mean
Value-only	6×10^{-5}	**50.3**	**11.3**	**93.4**	43.6	**71.2**	**27.6**	**49.6**
Attn. + MLP LoRA	5×10^{-5}	48.7	10.5	90.2	**44.2**	69.4	27.0	48.3

We compare the main Value-only SVP-V-GRPO run with an additional run that adapts attention and MLP projections through LoRA. The adapters use rank 16, scaling parameter 32, and zero dropout. They are attached to the fused QKV projection, attention-output projection, and both MLP projections in every GPT-2 block; all base parameters remain frozen.

Both runs follow the common GPT-2 recipe above, except that the LoRA learning rate is 5×10^{-5} rather than 6×10^{-5}. At epoch nine, Value-only updates achieve higher accuracy on five of six benchmarks (Table 4). The broader adaptation therefore offers no consistent advantage under the evaluated settings, although the learning-rate difference limits conclusions about update scope alone.

G Additional Semantic-Branching Cases

Figure 7 shows additional GSM8K-test problems. In both, the branch that reaches the correct answer appears at the intermediate computation the deterministic trajectory gets wrong, while the depths holding copied operands are shared across methods.

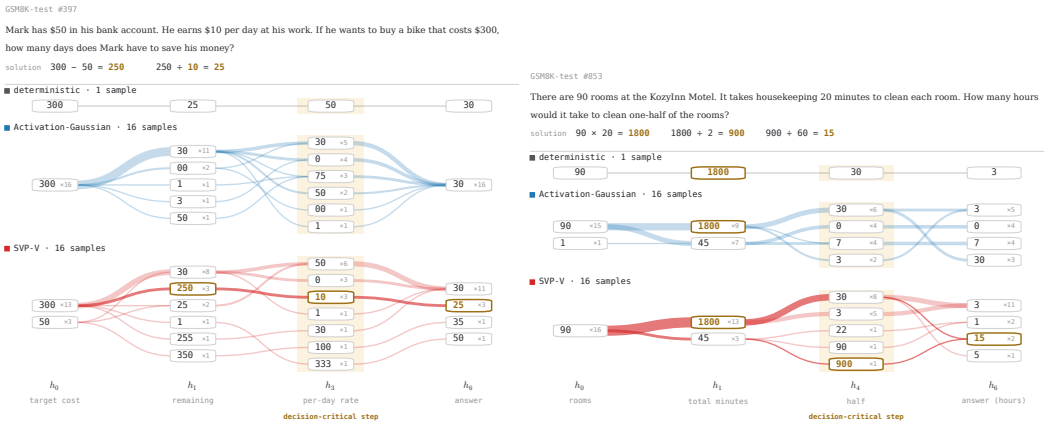

Figure 7: **Additional deterministic-failure cases with the decision-critical step at h_3 (left, GSM8K-test #397) and h_4 (right, #853).** Same readout tree as Figure 2; the shaded column is the intermediate computation the deterministic trajectory gets wrong. Only SVP-V samples read the gold value there (10 in $3/16$, 900 in $1/16$), and those samples are the ones that reach the correct answer ($3/16$ and $2/16$); no activation-Gaussian sample does.

H UMAP Visualization of Latent Trajectories

The LM-head tree in Figure 2(a) describes what the sampled states predict. We also visualize how the same trajectories are arranged in representation space.

H.1 Latent-Space Visualization Protocol

We use the same illustrative GSM8K failure case as Figure 2(a). For activation Gaussian noise and SVP-V, we collect 16 trajectories at their selected perturbation scales, together with the clean deterministic trajectory. For each displayed latent depth, we fit a separate two-dimensional UMAP projection jointly to the states from both methods and the clean run. Coordinates are therefore comparable within each panel but not across latent depths. Markers indicate whether the complete trajectory ultimately decodes the gold answer; this correctness label is attached only after answer decoding and is not used to fit the projection.

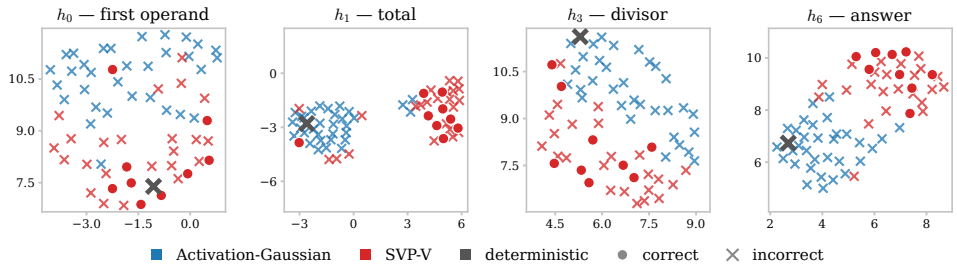

Figure 8: **Latent-space geometry for the failure case in Figure 2.** UMAP of 16 activation-Gaussian and 16 SVP-V trajectories at four latent depths. Circles and crosses indicate trajectories that ultimately decode correct and incorrect answers, respectively.

H.2 Geometry of the Sampled Trajectories

At the first intermediate computation, h_1, activation Gaussian states remain concentrated around the deterministic state, which reads out the incorrect total 22. The SVP-V states occupy a separate region that contains the trajectories reading out the correct total 24. A similar within-depth separation is visible in the later projections. By h_6, the SVP-V trajectories that decode the gold answer 6 form a region distinct from the failed deterministic trajectory, whereas the activation-Gaussian trajectories remain concentrated around incorrect outcomes.

This visualization is qualitatively consistent with successful samples departing from the neighborhood of the failed trajectory and following a different continuation. However, it is only an illustrative case. UMAP does not preserve all high-dimensional distances, and this figure is not benchmark-wide evidence that geometric separation always predicts correctness. The dataset-level JSD results in Section 4.2 provide broader evidence of readout variation, while the pass@16 results in Section 4.1 measure the resulting solution coverage.

I Spectral Allocation Ablation

We examine on COCONUT-GPT2 whether spectrum-proportional coefficient scales contribute to solution coverage beyond retaining the learned SVD directions. All variants keep the clean weight W and its singular-vector bases U, R unchanged, and differ only in the coefficient scales a of the additive perturbation

$$\Delta W = \alpha U \operatorname{diag}(a \odot \varepsilon) R^\top, \qquad \varepsilon \sim \mathcal{N}(0, I_r). \tag{21}$$

Full SVP uses $a = s$. The flat-spectrum control uses $a_i = \|s\|_2/\sqrt{r}$ for every component, removing spectrum proportionality. The permuted-spectrum control uses $a_i = s_{\pi(i)}$ for a fixed layer-wise random permutation π, breaking the correspondence between singular directions and their learned strengths while preserving the multiset of scales. Both controls preserve $\sum_i a_i^2 = \sum_i s_i^2$, and therefore match full SVP's expected squared Frobenius perturbation norm at the same α.

We fix $\alpha = 0.6$, the value previously selected for full SVP on COCONUT-GPT2, without control-specific retuning. Evaluation follows the main protocol: five unseen seeds (5–9), 64 trajectories per problem, and the unbiased pass@16 estimator on GSM8K-test.

Table 5: Spectral-allocation ablation at W_V on COCONUT-GPT2 using GSM8K-test (pass@16, five-seed mean $\pm$ standard deviation). All variants use the same pre-selected $\alpha = 0.6$ and match the expected squared Frobenius perturbation norm. The full-spectrum reference is reused from Table 1.

Spectral allocation	pass@16
Full spectrum	57.22 ± 0.48
Flat spectrum	55.94 ± 0.15
Permuted spectrum	56.05 ± 0.31

Table 6: Train-selected hyperparameters of Figure 4.

Model	Method	act.	W_Q	W_K	W_V	W_O	W_{up}	W_{down}
COCONUT	Top-p	$T = 1, p = 0.95$						
	SVP (α)	0.6	0.6	0.4	0.6	0.6	0.2	0.4
	Gaussian (σ)	0.15	0.1	0.1	0.06	0.05	0.05	0.05
CoDi-GPT2	Top-p	$T = 1.2, p = 0.95$						
	SVP (α)	0.7	0.7	0.4	0.5	0.6	0.2	0.4
	Gaussian (σ)	0.1	0.1	0.1	0.04	0.05	0.03	0.05

Full SVP improves pass@16 by 1.28 and 1.17 percentage points over the flat- and permuted-spectrum controls, respectively. These results support retaining the learned spectral allocation for solution coverage on COCONUT-GPT2 under this fixed-strength protocol.

J Hyperparameter Grids and Selected Values

All test-time hyperparameters are selected before evaluation. For each model–method pair, we select the hyperparameters that maximize five-seed mean pass@16 on the fixed 1,024-problem GSM8K training subset using seeds 0–4. For the location-grid experiment, we apply this procedure separately to each model–structure–location combination. We then hold the selected hyperparameters fixed across all evaluation benchmarks and evaluate using seeds 5–9. No evaluation benchmark is used for hyperparameter selection.

Table 6 reports the selected values for the location-grid experiment in Section C. Tables 7 and 8 list the search grids and selected hyperparameters for the stochastic methods evaluated in Tables 1 and 2, respectively.

Table 7: Per-model swept grids and the train-selected values behind Table 1.

Method	Swept grid	Selected
COCONUT		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1, p=0.95
Gaussian × Activation	$\sigma \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$	σ=0.15
SVP × Activation	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.6
Gaussian × W_V	$\sigma \in \{0.03, 0.04, 0.05, 0.06, 0.07\}$	σ=0.06
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.6
CoDi-GPT2		
Top-p	$T \in \{0.8, 1, 1.2, 1.5\} \times p \in \{0.9, 0.95, 1\}$	T=1.2, p=0.95
Gaussian × Activation	$\sigma \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$	σ=0.1
SVP × Activation	$\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$	α=0.7
Gaussian × W_V	$\sigma \in \{0.03, 0.04, 0.05, 0.06, 0.07\}$	σ=0.04
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5
CoDi-LLaMA-1B		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1, p=1
Gaussian × Activation	$\sigma \in \{0.005, 0.0075, 0.01, 0.0125, 0.015\}$	σ=0.01
SVP × Activation	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.4
Gaussian × W_V	$\sigma \in \{0.004, 0.005, 0.006, 0.007, 0.008\}$	σ=0.006
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5

Table 8: Per-model swept grids and the train-selected values behind Table 2.

Method	Swept grid	Selected
SIM-CoT-GPT2-COCONUT		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1.2, p=1
Gaussian × Activation	$\sigma \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$	σ=0.15
SVP × Activation	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5
SIM-CoT-GPT2-CoDi		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1.2, p=0.9
Gaussian × Activation	$\sigma \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$	σ=0.15
SVP × Activation	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.6
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5
SIM-CoT-LLaMA-1B		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1, p=0.95
Gaussian × Activation	$\sigma \in \{0.005, 0.0075, 0.01, 0.0125, 0.015\}$	σ=0.0075
SVP × Activation	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.4
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.5
SIM-CoT-LLaMA-3B		
Top-p	$T \in \{0.8, 1, 1.2\} \times p \in \{0.9, 0.95, 1\}$	T=1, p=0.9
Gaussian × Activation	$\sigma \in \{0.005, 0.0075, 0.01, 0.0125, 0.015\}$	σ=0.0125
SVP × Activation	$\alpha \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$	α=0.3
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.4
SIM-CoT-LLaMA-8B		
Top-p	$T \in \{0.8, 1, 1.2, 1.5\} \times p \in \{0.9, 0.95, 1\}$	T=1.2, p=0.95
Gaussian × Activation	$\sigma \in \{0.005, 0.0075, 0.01, 0.0125, 0.015\}$	σ=0.0075
SVP × Activation	$\alpha \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$	α=0.4
SVP × W_V **(SVP-V)**	$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	α=0.4