CHARACTERIZING LATENT REASONING DYNAMICS IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning in LLMs is typically explained through two distinct mechanisms: the autoregressive generation of discrete tokens, and the looped refinement of a continuous latent state. Both improve performance by increasing inference-time computation, yet the continuous computational dynamics underlying an off-the-shelf, token-based model's ordinary reasoning have not been directly observed. We bridge this gap by formalizing token-based reasoning as state transitions over KV-cache prefixes, compressing this high-dimensional space with a KV-cache VAE to map problem-specific error manifolds. Analyzing the resulting state-space dynamics, we find that standard autoregressive LLMs implicitly perform latent reasoning: their trajectories closely mimic gradient descent over basins. Specific tokens drive this descent, some rapidly exploiting the error gradient and others escaping local minima to explore, and trajectories quickly recover their descent path after off-policy perturbation. Applying these mechanistic insights, we causally improve inference-time scaling by optimizing the token-budget tradeoff between trajectory depth and breadth. Specifically, we demonstrate that initializing diverse latent "random seeds," much like in vanilla gradient descent, steers trajectories toward distinct local minima and yields a larger diversity in correct answers. Finally, we find that RL reshapes the latent space to contain fewer basins, along with faster and more likely convergence to basins correlated with improved performance.

1 INTRODUCTION

The success of reasoning in large language models (LLMs) is typically explained through two distinct mechanisms: the autoregressive generation of discrete tokens, i.e., chain-of-thought (CoT) reasoning (Wei et al., 2022; Nye et al., 2021; Yao et al., 2023), and the looped refinement of a continuous latent state (Zhu et al., 2025b; Hao et al., 2024; Wang et al., 2025; Jolicoeur-Martineau, 2025; Geiping et al., 2025). Both improve performance by increasing inference-time computation, yet how they relate remains unresolved.

One line of work questions whether intermediate tokens are faithful traces of computation at all, arguing they instead act as a learned scaffold (Pfau et al., 2024; Valmeekam et al., 2025; Kambhampati et al., 2026; Baherwani et al., 2026). A parallel line shows that models can instead be trained to refine an explicit continuous state by descending a learned energy or attractor basin (Du et al., 2024; Chen et al., 2025b; Huang et al., 2026; Jolicoeur-Martineau, 2025), but this evidence comes entirely from purpose-built recurrent architectures or models retrained specifically for continuous-space reasoning. Crucially, no existing work has studied the continuous computational dynamics underlying an off-the-shelf, token-based LLM during its ordinary reasoning.

We bridge this gap by formalizing token-based reasoning as a continuous trajectory of state transitions, where each state is the model's KV-cache prefix at a given generation step. Because this state space is too high-dimensional to analyze directly, we project it into a tractable, low-dimensional coordinate system using a purpose-trained KV-cache VAE. Within this compressed space, a prompt initializes a question-conditioned stochastic dynamical system, and a single chain of thought traces out one realized trajectory of that system. To characterize the underlying mechanics, we measure the latent trajectory against two complementary fields: a depth-based *error field* (the final-answer error rate estimated via multiple rollouts from a given state) and a *residual field* (the rate and direction of successive latent transitions). Additionally, we study the impacts of RL by analyzing the changes in

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.

latent characteristics, and find mechanistic explanations beyond policy sharpening. After training on over 1M traces and analyzing over 100k trajectories across three reasoning models—DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-4B, and Nemotron-Nano-4B—we present three primary contributions:

- **Latent gradient descent in token-based LLMs:** The reasoning dynamics of a standard, unmodified LLM are well approximated by gradient descent over a problem-specific error surface.
- **Token-level control of the descent:** The model's own vocabulary steers this process where specific phrases systematically align the trajectory with the local error gradient to exploit a solution path, while others move against the gradient to escape a local minimum and explore alternatives.
- **Causal, inference-time control of token budget:** Token injection and direct latent perturbation confirm these dynamics are causal, not merely correlational. Seeding trajectories at different points in latent space, akin to random restarts in gradient-based optimization, drives the model to distinct local minima, improving the depth–breadth tradeoff for a fixed token budget and, consequently, pass@k performance.

2 Related Work

Reasoning in large language models has been studied through two disjoint lenses: the semantic fidelity of discrete token generation, and the continuous, iterative refinement of latent states. We sit at their intersection, building on token skepticism, purpose-built latent architectures, and the internalized optimization capacity of transformers.

Faithfulness of CoT: A growing body of work challenges the assumption that intermediate CoT tokens reflect a model's true internal algorithm. Trace validity is only weakly coupled to final-answer correctness, and replacing reasoning traces with semantically unrelated text can cause little to no accuracy loss Pfau et al. (2024); Baherwani et al. (2026); Valmeekam et al. (2025). CoT length tracks training-set statistics rather than problem difficulty, prompting explicit warnings against anthropomorphizing intermediate tokens Kambhampati et al. (2026). Together, these results cast the generated CoT tokens primarily as a mechanism for "prompt augmentation" expanding the forward-pass compute, rather than an algorithmic report. *We take this skepticism as our starting point: if tokens are unfaithful traces, the underlying computation must be measured directly in latent space.*

Latent and Non-Lexical Reasoning Paradigms: A parallel line bypasses token-level unfaithfulness by training models to reason directly in continuous space. Coconut Hao et al. (2024) feeds the final hidden state back as the next input embedding, yielding a superposition of reasoning paths. Chain-of-Superposition operates over convex combinations of vocabulary embeddings Deng et al. (2026); Zhu et al. (2025a); BoLT treats visible text as a discrete trace of a latent thought process refined via importance reweighting Ruan et al. (2025). Closest to our substrate, KaVa distills a compressed teacher KV-cache into a student, making the KV-cache itself the object generated during latent reasoning Kuzina et al. (2026). These paradigms establish that non-lexical reasoning is viable, and that the KV-cache specifically is a usable substrate. However, each technique relies on a custom architecture or a continuous-space retraining objective. They do not measure the latent dynamics already present in a standard, unmodified model. *We address this directly: our KV-cache VAE compresses the prefix of an off-the-shelf LLM into a low-dimensional trajectory, with no retraining or change to generation.*

Iterative Refinement, Attractors, and Energy Landscapes: A related line of work models reasoning as an iterative refinement over an energy landscape. Tiny Recursion Models (TRM) replace CoT with explicit architectural recursion, outperforming much larger models on combinatorial puzzles through continuous latent updates Jolicoeur-Martineau (2025); Wang et al. (2025). Equilibrium Reasoners frame weight-tied recurrent models as dynamical systems whose fixed-point residuals proxy for correctness across distinct attractor regimes Huang et al. (2026). EBM-CoT optimizes latent embeddings via Langevin dynamics on a learned energy surface Chen et al. (2025b). These works extend classical dynamical-systems analyses to purpose-built recurrent solvers, establishing energy-based descent as a viable reasoning mechanism, but only within architectures built for it.

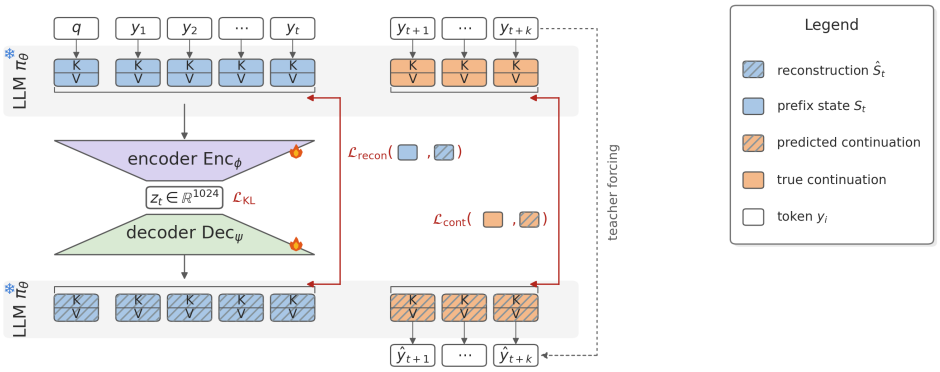

Figure 1: **The KV-cache VAE.** *Top:* the frozen model answers a question, denoted by the prompt q and writes reasoning tokens y_0 to y_{t+k}. Every token leaves a key-value entry, and the first t entries after the prompt form the state S_t (blue). *Middle:* the trainable encoder attends over every entry of S_t and returns the posterior mean z_t; the decoder broadcasts z_t back to every position to reconstruct a cache $\hat{S}_t$ (hatched). *Bottom:* the same frozen model is run from $\hat{S}_t$ on the trace's own next k tokens (teacher forcing), and the cache it writes is compared with the true continuation cache (orange). Training minimizes a KL term on z_t ($\mathcal{L}_{\text{KL}}$), the reconstruction error ($\mathcal{L}_{\text{recon}}$), and a continuation error ($\mathcal{L}_{\text{cont}}$) which encodes the part of S_t that the trajectory carries forward.

Independent evidence also support the premise that correctness of an answer in-progress is linearly decodable from a reasoning model's hidden state (Zhang et al., 2025). This is precisely the latent correctness signal a measurable error or residual field would need to exist. However, none of these works explore whether a standard, non-recurrent model exhibits comparable dynamics unprompted. *We show it does: an unmodified model's ordinary token-based CoT approximates gradient descent over a problem-specific error surface, combined with basin-hopping between local minima, measured via speed and direction fields in our compressed latent space.*

Mesa-Optimization and Internalized Gradient Descent: Theoretical work shows transformers can internalize an optimization algorithm within a single forward pass. Trained attention layers can execute single-step or multi-step preconditioned gradient descent for in-context linear regression Garg et al. (2022); Akyürek et al. (2022); Ahn et al. (2023); von Oswald et al. (2022; 2023). Looped transformers and linear-attention recurrences have been proven to implement multi-step gradient descent over context Gatmiry et al. (2024); Huang et al. (2025). These results confirm iterative computation can implement gradient descent, but only in linear or synthetic settings, disconnected from ordinary generation in real models. *We bridge this gap by providing empirical evidence that ordinary token-based CoT implements latent iterative descent, connecting theoretical results on internalized optimization to reasoning in real LLMs.*

3 Methods

3.1 Reasoning as a dynamical system on the KV prefix

We study reasoning as a dynamical system where each trajectory corresponds to a single CoT. For a prompt q, let $y_{1:t}$ denote the first t tokens generated after the `<think>` tag, and let T be the index at which the model emits `</think>`. We index time by t, so that $t = 0$ is the state after the prompt, chat template, and opening tag have been processed, but before any reasoning token is written. The *state* of an autoregressive transformer at time t is its KV cache over the prefix,

$$S_t = (K_{\le t}, V_{\le t}) \in \mathbb{R}^{(|q|+t)\times D_{\text{KV}}}, \qquad D_{\text{KV}} = 2\,L\,H_{\text{kv}}\,d_{\text{head}} \tag{1}$$

where L is the number of layers, H_{kv} the number of key–value heads and d_{head} the head dimension. S_t is a *Markov state*: in a standard LLM, the next-token distribution, and hence everything the model will do next, is a function of S_t alone. One autoregressive step samples a token $y_{t+1} \sim \pi_\theta(\cdot \mid S_t)$

and appends its keys and values,

$$S_{t+1} = F_\theta(S_t, y_{t+1}), \qquad y_{t+1} \sim \pi_\theta(\cdot \mid S_t) \tag{2}$$

where F_θ runs one forward pass of the frozen model on y_{t+1} and appends the resulting keys and values to the cache. We write $F_\theta(S_t, y_{t+1:t+k})$ for k such steps in sequence, so that $S_{t+k} = F_\theta(S_t, y_{t+1:t+k})$ for any cache S_t and token sequence.

A prompt therefore selects a question-conditioned stochastic dynamical system on the space of prefixes, and a single chain of thought is one trajectory $S_0, S_1, \ldots, S_T$ of that system where token sampling introduces stochasticity in the flow. We call the dynamics inside the `<think>` region *reasoning*, and restrict every analysis in the paper to it.

A fixed-dimensional coordinate. The difficulty with S_t as a state is that it grows with t, so it is extremely high dimensional and there is no clear shared coordinate system to analyze. We therefore need to learn a compression $z_t = \mathrm{Enc}_\phi(S_t) \in \mathbb{R}^d$, where $d = 1024$ and study the dynamics in the coordinate z. All fields, speeds and directions in this paper are measured on z_t.

3.2 The KV-cache VAE

Why a VAE? We require the following desiderata for encoding $z_t = \mathrm{Enc}(S_t)$. **(i) Past encoding.** A decoder recovers the full cache prefix, $\mathrm{Dec}(z_t) \approx S_t$, so a latent can be written back into the model. **(ii) Future encoding.** $P(S_{>t} \mid S_t) \approx P(S_{>t} \mid \mathrm{Dec}(z_t))$ the dynamics on z are *Markov*, and the probability of a correct answer becomes a function of z_t alone. Compression is lossy and reconstruction discards information by norm distance, not by relevance, so (ii) does not follow from (i). Here we explicitly try to ensure that information that guides the trajectory is not discarded. **(iii) Geometry encoding.** Latents should be defined over a smoothed surface so that nearby latents decode to plausibly nearby KV-caches.

We address these conditions by **(i)** a KV-cache reconstruction loss term **(ii)** a continuation loss term which runs the frozen model from $\mathrm{Dec}(z_t)$ and compares its next k steps with the trace's own (Section 3.2) **(iii)** a KL term trains the decoder on a neighbourhood of every z_t rather than on the point alone. Thus, our encoder should be transformer-architecture based to attend over the whole prefix, since a statistic sufficient for (i) or (ii) must see all of S_t. And we use a VAE to ensure the neighborhood around z_t are also meaningful KV-caches.

Why not use text or a hidden state? Two cheaper summaries of the state are commonly used rather than an autoencoder. The first is the text $y_{1:t}$ itself. Text is a very lossy summary of S_t, and each token does not capture the state of the full trace. Additionally a significant body of work has warned that chain of thought is not a faithful account of the computation that produced it (Turpin et al., 2023; Chen et al., 2025a; Palod et al., 2026). The second commonly used summary state is a single late-layer hidden state $h_t^{(\ell)}$ at the current token position t. This state is richer than tokens because self-attention in all the previous and current layers has already mixed the information from the prefix into the hidden state. However, the next-token prediction objective only requires $h_t^{(\ell)}$ to carry information to optimize for the next token; information the model will attend to later can remain at earlier positions and never pass through it (Ustaomeroglu et al., 2026). Consistent with this, in Figure 4 the dynamics of $h_t^{(\ell)}$ show none of the structure that the full-state coordinate does. Both proxies assume in advance that the part of the model's latent computation that affects the rest of the trace happens to be stored in one vector or one string. We do not want our conclusions to rest on that assumption, so we instead learn a compression of the full state under an explicit objective.

Architecture. The encoder and decoder (Dec_ψ) are 6-layer bidirectional Transformers. The encoder is fed the full KV cache S_t. A learned CLS token aggregates the prefix, and linear heads on its output give the posterior

$$q_\phi(z \mid S_t) = \mathcal{N}\big(\mu_\phi(S_t),\, \mathrm{diag}\, \sigma^2_\phi(S_t)\big), \qquad \mu, \log \sigma^2 \in \mathbb{R}^{1024}. \tag{3}$$

We take the posterior mean as the latent state,

$$z_t := \mu_\phi(S_t), \tag{4}$$

so the coordinate is a deterministic function of the prefix. The decoder Dec_ψ broadcasts one latent to all cached positions and reconstructs a full cache $\hat{S}_t$ of the same shape as S_t.

Training objective. The loss is a β-VAE on the cache plus a continuation term, $\mathcal{L}_{\text{cont}}$,

$$\mathcal{L} = \underbrace{\|\hat{S}_t - S_t\|^2}_{\mathcal{L}_{\text{recon}}} + \beta \underbrace{\text{KL}\big(q_\phi(z \mid S_t) \,\|\, \mathcal{N}(0, I)\big)}_{\mathcal{L}_{\text{KL}}} + \mathcal{L}_{\text{cont}}, \tag{5}$$

where $\hat{S}_t = \text{Dec}_\psi(z_t)$. The first two terms address desiderata (i) and (iii). For (ii), we split the continuation loss into two terms:

$$\mathcal{L}_{\text{cont}} = \gamma_{\text{tf}} \underbrace{\big\|F_\theta(\hat{S}_t, y_{t+1:t+k}) - S_{t+k}\big\|^2}_{\mathcal{L}_{\text{tf}}} + \gamma_{\text{gen}} \underbrace{\big\|F_\theta(\hat{S}_t, \hat{y}_{t+1:t+k'}) - S_{t+k'}\big\|^2}_{\mathcal{L}_{\text{gen}}} \tag{6}$$

The two terms differ only in where the continuation tokens come from. The teacher-forced term, $\mathcal{L}_{\text{tf}}$, takes the trace's original k tokens $y_{t+1:t+k}$. The generative term, $\mathcal{L}_{\text{gen}}$, lets the model autoregressively generate k' tokens $\hat{y}_{t+1:t+k'}$ greedily starting from $\hat{S}_t$, with a straight-through estimator. We tried different variants of continuation loss before arriving on the above terms (Appendix A.1).

3.3 VAE VALIDITY

Before analyzing dynamics in z, we check that the trained VAE meets desiderata (i)–(iii). For most metrics we check against a *resampled* baseline: another trace of the same problem, or a resampled continuation of the same prefix, which is the natural scale of variation the encoder must beat. The decoded cache $\hat{S}_t$ should be closer to S_t than another trace is at the same depth; the frozen model run from $\hat{S}_t$ should track the true continuation more closely than a resampled one; and generating from $\hat{S}_t$ should *solve the problem about as often as the true prefix does*. Finally, since z_{t+1} is a fresh encoding of S_{t+1} rather than a function of z_t, we test closure directly: states from different traces that pass close in z should diverge over the next o tokens no faster than two resamples of an identical prefix, and far slower than unrelated states at the same depth. We find this holds (Appendix A.2), so we treat z as an approximate state. Full results are in Appendix A.2, Table 8.

3.4 TWO MANIFOLDS OVER THE LATENT

Latent reasoning models are trained so that their state descends an energy, $\dot{x} = -\nabla E(x)$, refining a solution and settling at a minimum. With z validated as an approximate state, we explore whether the observed dynamics of a token model are consistent with the same picture. To make the comparison concrete, we define an *error landscape*, the task error the model would incur if it stopped at z, and a *flow*, how the model actually evolves z. The equilibrium claim is that the two coincide: the flow points down gradients and slows at minima.

The error field. We measure the energy of a state S_t as how often the model answers wrongly if reasoning stops there: append `</think>`, sample the answer 16 times at temperature 0.6, and grade the boxed answer,

$$E(S_t) = 1 - P_\theta\big(\text{correct} \mid \text{force close at } S_t\big). \tag{7}$$

Probing a dense grid of depths along every trace gives, for each problem, a scattered sample $\{(z_i, E_i)\}$ over its own state cloud, and every statistic below is computed from that sample. For visualization only, we project a problem's states onto their top two principal components and smooth E with a Gaussian kernel.

Comparing speed. Latents are stored every $\Delta = 4$ reasoning tokens. For a step from z_t to $z_{t+\Delta}$ we write

$$\Delta z_t = z_{t+\Delta} - z_t, \qquad v_t = \|\Delta z_t\|/\Delta, \qquad u_t = \Delta z_t / \|\Delta z_t\| \tag{8}$$

for the displacement, speed and unit direction. We test if v_t is low when $E(S_t)$ is low, especially across traces at a similar depth t.

Comparing direction. Our hypothesis also predicts that u_t points down E, which requires ∇E at the step. The samples $\{(z_i, E_i)\}$ are sparse, so we compute an estimate of the gradient locally. States are projected onto the top $k = 32$ principal components of the problem's own cloud (98% of

variance), and at the step midpoint $x_t = \frac{1}{2}(z_t + z_{t+\Delta})$ we fit a plane to the probed values near x_t by kernel-weighted least squares,

$$(\hat{a}, \nabla\hat{E}(x_t)) = \arg\min_{a,\,b} \sum_j w_j \big(E_j - a - b^\top (z_j - x_t)\big)^2, \qquad w_j = \exp\big(-\|x_t - z_j\|^2 / 2h^2\big), \quad (9)$$

taking the fitted slope as the local gradient. With uniform weights this is the global linear fit of E on z; the kernel makes it local. The bandwidth h is twice the median nearest-neighbour distance among the probed states, so it adapts to each problem's cloud. Traces used to estimate gradients are held out from any analysis. The alignment of a step with descent is

$$a_t = \cos\big(u_t, -\nabla\hat{E}(x_t)\big), \quad (10)$$

We check if and when the cosine alignment is high.

3.5 Interventions

The measurements defined thus far are all observational. To check that the interpretation they support is causal, we intervene in two ways, and observe how trajectories evolve: *token injections* perturbs states by sampling tokens off policy *latent perturbations* moves the state in a direction in latent space.

Token injections. In Eq. 2 the policy π_θ chooses the token which influences the next state, so the policy is one way to control the direction of the flow. To test how much of the observed direction is controlled by the policy's sampled tokens, we replace tokens over a given window. A parent trace is cut at depth t_c, the next $W = 64$ tokens are drawn from a substitute sampler ν,

$$\tilde{y}_{t_c+1:t_c+W} \sim \nu, \qquad S_{t_c+W} = F_\theta\big(S_{t_c}, \tilde{y}_{t_c+1:t_c+W}\big), \quad (11)$$

after which tokens go back to being generated by π_θ for the rest of the trace. Perurbation sources range from the model's own policy at higher temperatures, to tokens copied from another problem's trace at the same fractional depth (fluent but about the wrong problem), to tokens drawn uniformly from the vocabulary (no semantics at all). We measure how the dynamics change when the sampling model F_θ is increasingly off policy.

Latent perturbation. To perturb in latent directions, we add only the change the decoder attributes to the displacement, to minimize capturing changes in KV from reconstruction error.

$$S_t \;\leftarrow\; S_t + \big[\mathrm{Dec}_\psi(z_t + \delta v) - \mathrm{Dec}_\psi(z_t)\big], \quad (12)$$

Here v is a unit direction and δ a displacement size in latent units. Directions are drawn inside the top $k = 8$ principal components of the global state cloud. In a non-convex energy landscape, different initializations converge to distinct local minima. We generate B traces with perturbations to their initial KV entries which we denote as **Latent seeds** to enforce variants of gradient descent algorithms.

4 Results

4.1 Direction: the policy sets the direction of the flow

Averaged over all steps, the flow descends E. This is expected, since reasoning raises accuracy so E must fall along the traces. The informative question is what produces the alignment. If intermediate tokens are units of compute whose content matters little (Pfau et al., 2024; Valmeekam et al., 2025), then W tokens from any source should move the state the same way. Following Eq. 11, we replace π_θ by a substitute token sampler ν for a window of $W = 64$ tokens and then return control to π_θ.

Gradient alignment falls when sampling off policy tokens. Figure 2a shows a_t during the perturbation window. Tokens sampled from π_θ keep descending, as much at temperature 1.0 as at 0.6, and about half as much at 1.5. Every other source lands at chance, whether the tokens are fluent reasoning copied from another problem, high-entropy draws restricted to the policy's own top-50 candidates, or completely random tokens. After the window every arm returns to the control's descent rate within about 100 tokens. We confirm that every arm's window states sit in regions the field was fit on, so the gradients remain well defined over the perturbation region. Thus, replace the policy and the descent stops, restore it and the descent resumes from wherever the state now is.

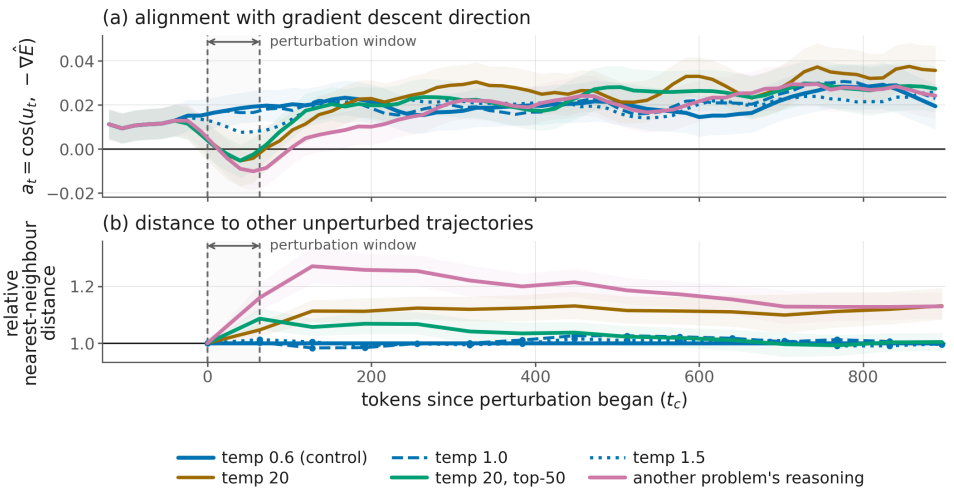

Figure 2: **How token perturbations displace the reasoning state.** Traces are perturbed by splicing in 64 tokens from various sources. The x-axis is reasoning tokens since the perturbation window began. **(a)** Alignment a_t of each latent step with the local error gradient (Eq. 10). Lower temperature tokens descend the gradient, while content-irrelevant tokens move orthogonal to the gradient. **(b)** Distance to the nearest unperturbed trajectories at matched depth, relative to the matched control (Eq. 13); 1 means indistinguishable from an unperturbed trace. Noise from the policy's own vocabulary is forgotten within a few hundred tokens while wrong-problem reasoning and random tokens leave the state permanently displaced.

Alignment resumes with the policy, although the state may have changed. Although every arm descends again after the perturbation, they do not perform equally. Temperatures 1.0, 1.5 and the top-50 arm match the controls final accuracy, while another problem's reasoning and random tokens perform .08 worse.

Figure 2b explains why. For a perturbed trace with sampler ν we measure, at each depth t, how far its state sits from the problem's unperturbed trajectories $r \in R$,

$$d_\nu(t) = \frac{1}{10} \sum_{r \in \mathrm{NN}_{10}(t)} \|z_t^\nu - z_t^r\|, \qquad \text{plotted as } d_\nu(t)/d_{\mathrm{ctrl}}(t), \tag{13}$$

a value of 1 means the state is as close to normal trajectories as a normal trajectory is.

Tokens the policy would never write disrupt the continuation of the flow by polluting the KV cache. Although the state doesn't move much, because random changes to the KV cache are not absorbed by the encoder, by the end of the perturbation window (1.04), the flow is disrupted and it does not return to the unperturbed region of states.

Tokens from another trace perturb the trajectory the most. Tokens come from the policy and carry reasoning content that is kept in the state. So the reasoning state has genuinely changed and the model continues from a confused position.

In contrast, nonsense tokens that come from the models policy neither pollute the KV cache, nor get absorbed in the reasoning state, and so the flow remains undisturbed and they return back to the same region as non-perturbed states.

Where final accuracy reports one number per arm, the trajectory around the perturbation shows when and how the state was changed, separating content from noise. Only on-policy, on-topic tokens make progress; in-support content-free tokens are harmless; wrong-topic and out-of-support tokens permanently corrupt the state.

Controlling exploration and exploitation. We find that the model has leveraged particular token semantics to control which direction to move in. Scoring each step by the phrase it contains, some phrases such as *think again* systematically travel up gradients to explore (climbs on 71% of its steps), while others like *hence* travel down gradients (on 80% of steps) exploiting (Figure 3).

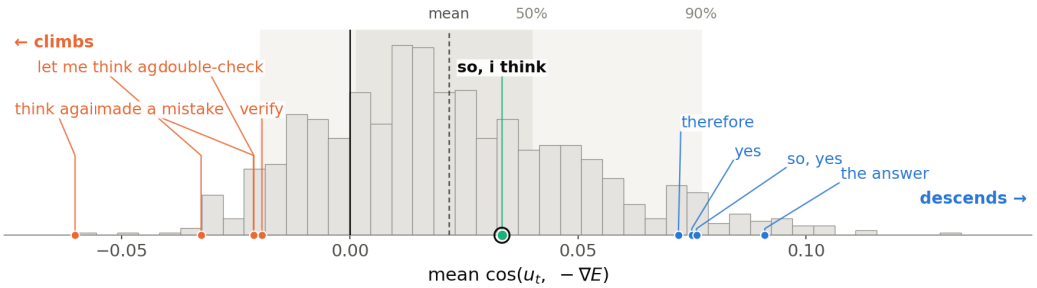

Figure 3: **Phrases that most consistently move the state up or down gradients.** For each phrase, the average direction of the latent step is compared to the local gradient, across problems. Most steps go downhill and the phrase *so, i think* is near average. Checking phrases go uphill, committing phrases downhill. Shown are a few of the phrases with the largest average effect in either direction.

Thus, the model can ***(i)*** *check where it is*: intermediate-answer correctness $E(S_t)$ is linearly decodable from its internals (Zhang et al., 2025); ***(ii)*** *Decide when to backtrack*: backtracking is steered by a linear direction in activation space (Venhoff et al., 2025); **(iii)** And from our result, backtracking moves the state up ∇E and committing movest down it. The model therefore has every mechanism of a simple feedback controller for exploration and exploitation; whether it wires them together is not established here.

4.2 Speed: the flow slows with depth, not with the landscape

The speed prediction fails. Over a problem's states the latent speed v_t forms a smooth field that resembles the error field, while a raw hidden state's speed is structureless (Figure 4), but the resemblance is carried by depth. Fitting $\log v_t$ on $\log t$ alone predicts speed out of sample with $R^2 = 0.79$.

This is consistent with the observation that these models overthink and do not stop when they have the answer (Zhang et al., 2025).

Figure 13 shows the consequence for test-time scaling: if speed tracked the landscape, its contours over depth and breadth would follow those of best-of-B error, which fall steeply with B as more restarts find a solution. Instead the speed contours are nearly flat in B; every trace slows on the same schedule whether or not it has arrived.

It is not necessarily true that the speed of latent updates should slow down as the model becomes more confident. It is entirely possible that model confidence has no relation to latent speed. However, if we want reasoning models to behave more like latent reasoning models, this could be part of the objective, and doing so may improve inference time scaling.

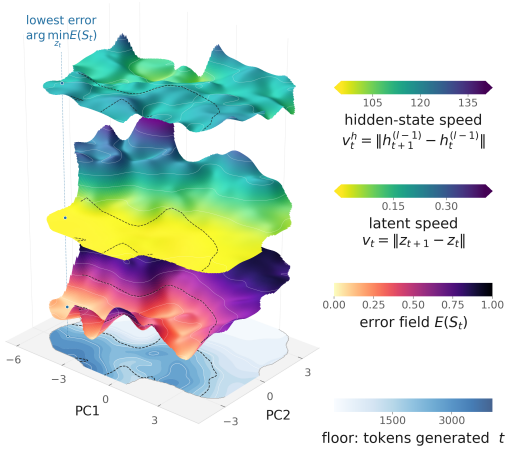

Figure 4: **Four fields for a single problem**, over the top two PCA components (43% variance) of the latent state z_t. Bottom to top: (i) tokens generated t; (ii) error rate when the answer is forced, 16 resamples; (iii) latent speed v_t; (iv) hidden-state speed v_t^h. Latent speed is smooth and correlated with error, $\rho(v_t, E_t) = +0.75$, and with depth, $\rho(v_t, t) = -0.96$. Using the hidden-state as a latent, the speed is correlated with neither, $\rho(v_t^h, E_t) = -0.22$, $\rho(v_t^h, t) = +0.28$ (same fig in its own coordinates, Fig. 12).

4.3 Causal Intervention Algorithms

We sweep over temperatures and hyperparameters in Appendix 4.3, and Table 1 reports the best results between standard temperature sampling and latent seeding. If a reasoning trajectory descends a landscape with several basins, two things should hold: different starting points reach different

answers, and pushing traces to different basins reach different answers. Both hold under latent seeding explained in Methods 3.5 (Table 1). Specifically, *latent seeds* at $\delta = 8$ increase both distinct answers and performance. Because restarts begin in different basins, the value of breadth, $\text{pass@8} - \text{pass@1}$, stays flat while it falls consistently under temperature sampling (Figure 5).

Table 1: Comparison of baseline temperature sampling versus latent seeds and across models.

		Baseline ($T = 0.6$)			**Latent Seeds**			
Model	**Dataset** (n)	**pass@1**	**pass@8**	**distinct**	**pass@1**	**pass@8**	**distinct**	**Δ breadth**
Qwen3-4B	MATH-500 (32)	**0.869**	0.943	1.43	0.867	**0.957**	**1.50**	+0.015
	AMC23 (40)	**0.832**	0.943	**1.51**	0.805	**0.948**	1.45	+0.033
	GSM8K (100)	**0.929**	0.963	1.12	0.896	**0.980**	**1.22**	**+0.050**
Nemo-Nano	MATH-500 (32)	0.788	0.903	1.71	**0.835**	**1.000**	**2.33**	+0.028
	AMC23 (40)	0.782	0.923	1.43	**0.821**	**0.928**	**1.58**	−0.038
	GSM8K (100)	**0.907**	**0.967**	**1.25**	0.814	0.899	1.22	+0.024
R1-1.5B	MATH-500 (32)	**0.636**	0.733	2.13	0.535	**0.812**	**3.00**	+0.079
	AMC23 (40)	**0.692**	0.916	2.57	0.654	**0.942**	**3.02**	+0.078
	GSM8K (100)	**0.779**	0.954	2.30	0.734	**1.000**	**2.78**	+0.091

4.4 Latent Dynamics of Reinforcement Learning

We train Qwen3-4B with GRPO on 2,048 problems from the MATH training split, With batch size 16 problems and group size 32 using constant learning rate 10^{-6}. Further details are in Appendix G

We re-train the VAE in the identical fashion on the DeepMath traces as before for several checkpoints, and analyzed resulting basins on 64 sampled MATH500 problems.

Table 2: **Basins and generation metrics across RL training.**

checkpoint	pass@1	basins	largest basin	final states in a basin	median reasoning length
base	0.374	3.00	0.16	55%	2,632
step 31	0.498	2.65	0.17	61%	2,222
step 95	0.743	2.57	0.18	63%	1,632
step 127	0.773	2.52	0.18	68%	1,496

We find that over RL steps, an increasing percent of final states end up in a basin, as the accuracy grows. The quantity of basins steadily decreases, indicating the sharpening of the distribution, and fewer local minima, which could explain improved performance, and allow for sharper traversal over the latent space. The model also makes a more efficient search over the latent space, requiring a far lesser median reasoning length over training, indicating that the traversal itself has been optimized.

5 Conclusion

By framing a language model's chain-of-thought as a dynamical system on its KV cache and compressing this high-dimensional state via a trained VAE, we demonstrate that standard token-based LLMs implicitly perform continuous latent reasoning, actively steering their trajectories with the model's own vocabulary toward attractor basins. RL training sharpens this landscape directly: as accuracy improves over training, the number of attractor basins steadily decreases and reasoning trajectories shorten, indicating a distribution that is not only more accurate but easier to traverse. Direct latent perturbations extend this picture causally, letting us trade trajectory depth for breadth under a fixed token budget and improve pass@k. Together, these findings reveal that the discrete generation of token-based models already implements the iterative refinement targeted by specialized latent-reasoning architectures, suggesting that reasoning in language and latent space are far less distinct paradigms than their differing designs imply.

AI USE STATEMENT

This paper studies large language models as the primary subject of research. The proposed method trains a KV-cache VAE to compress and analyze the latent reasoning dynamics of frozen, off-the-shelf language models, and performs causal interventions (token injection, latent perturbation) on their reasoning trajectories, as described in the main text. In addition, the authors used large language model tools to assist with non-substantive writing tasks, including improving clarity, editing grammar, drafting checklist responses, and revising explanatory text. All technical ideas, algorithmic design choices, theoretical claims, proofs, experimental results, tables, and conclusions were produced, checked, and approved by the authors. The authors take full responsibility for the content of the paper, including any text that was edited with LLM assistance.

ETHICS STATEMENT

This work studies the latent reasoning dynamics of frozen, off-the-shelf Transformer language models using existing pre-training and evaluation datasets. We do not introduce new datasets containing personal or sensitive information, nor do we conduct experiments involving human subjects. The broader ethical considerations of the resulting language models are similar to those of other pre-trained language models of comparable scale.

REPRODUCIBILITY STATEMENT

We provide the architectural specification, model configurations, training budgets, optimization setup, and evaluation protocol needed to reproduce our experiments. Additional implementation and experimental details are provided in the Appendix. We plan to release the code used to train and evaluate the VAE model, along with our model checkpoints.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning, 2023. URL http://arxiv.org/abs/2306.00297. arXiv:2306.00297 [cs].

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models, 2022. URL http://arxiv.org/abs/2211.15661. arXiv:2211.15661 [cs].

Vatsal Baherwani, Tom Goldstein, and Ashwinee Panda. Not all llm reasoning is visible in the chain-of-thought, 2026. URL https://arxiv.org/abs/2607.22925.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think, 2025a. URL https://arxiv.org/abs/2505.05410.

Zhikang Chen, Sen Cui, Deheng Ye, Yu Zhang, Yatao Bian, and Tingting Zhu. Think consistently, reason efficiently: Energy-based calibration for implicit chain-of-thought, 2025b. URL https://arxiv.org/abs/2511.07124.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition, 2026. URL https://arxiv.org/abs/2510.15522.

Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning Iterative Reasoning through Energy Diffusion, June 2024. URL http://arxiv.org/abs/2406.11179. arXiv:2406.11179 [cs].

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes, 2022. URL http://arxiv.org/abs/2208.01066. arXiv:2208.01066 [cs].

Khashayar Gatmiry, Nikunj Saunshi, Sashank Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach, February 2025. URL http://arxiv.org/abs/2502.05171. arXiv:2502.05171 [cs].

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, December 2024. URL http://arxiv.org/abs/2412.06769. arXiv:2412.06769 [cs].

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488.

Jianhao Huang, Zixuan Wang, and Jason D. Lee. Transformers learn to implement multi-step gradient descent with chain of thought, 2025. URL https://arxiv.org/abs/2502.21212.

Alexia Jolicoeur-Martineau. Less is More: Recursive Reasoning with Tiny Networks, October 2025. URL http://arxiv.org/abs/2510.04871. arXiv:2510.04871 [cs].

Subbarao Kambhampati, Karthik Valmeekam, Siddhant Bhambri, Vardhan Palod, Lucas Saldyt, Kaya Stechly, Soumya Rani Samineni, Durgesh Kalwar, and Upasana Biswas. Position: Stop anthropomorphizing intermediate tokens as reasoning/thinking traces!, 2026. URL https://arxiv.org/abs/2504.09762.

Anna Kuzina, Maciej Pioro, Paul N. Whatmough, and Babak Ehteshami Bejnordi. Kava: Latent reasoning via compressed kv-cache distillation, 2026. URL https://arxiv.org/abs/2510.02312.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114, 2021. URL https://arxiv.org/abs/2112.00114.

Vardhan Palod, Upasana Biswas, and Subbarao Kambhampati. Evaluating the false trust engendered by llm explanations, 2026. URL https://arxiv.org/abs/2605.10930.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's Think Dot by Dot: Hidden Computation in Transformer Language Models, April 2024. URL http://arxiv.org/abs/2404.15758. arXiv:2404.15758 [cs].

Yangjun Ruan, Neil Band, Chris J. Maddison, and Tatsunori Hashimoto. Reasoning to learn from latent thoughts, 2025. URL https://arxiv.org/abs/2503.18866.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting, 2023. URL http://arxiv.org/abs/2305.04388. arXiv:2305.04388 [cs].

Muhammed Ustaomeroglu, Baris Askin, Gauri Joshi, Carlee Joe-Wong, and Guannan Qu. Internal Planning in Language Models: Characterizing Horizon and Branch Awareness, February 2026. URL http://arxiv.org/abs/2509.25260. arXiv:2509.25260 [cs] version: 2.

Karthik Valmeekam, Kaya Stechly, Vardhan Palod, Atharva Gundawar, and Subbarao Kambhampati. Beyond Semantics: The Unreasonable Effectiveness of Reasonless Intermediate Tokens, November 2025. URL http://arxiv.org/abs/2505.13775. arXiv:2505.13775 [cs].

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding Reasoning in Thinking Language Models via Steering Vectors. March 2025. URL https://openreview.net/forum?id=OwhVWNOBcz&referrer=%5Bthe%20profile%20of%20Philip%20Torr%5D(%2Fprofile%3Fid%3D~Philip_Torr1).

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2022. URL http://arxiv.org/abs/2212.07677. arXiv:2212.07677 [cs].

Johannes von Oswald, Eyvind Niklasson, Maximilian Schlegel, Seijin Kobayashi, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Blaise Aguera y Arcas, Max Vladymyrov, Razvan Pascanu, and João Sacramento. Uncovering Mesa-Optimization Algorithms in Transformers, 2023. URL http://arxiv.org/abs/2309.05858. arXiv:2309.05858 [cs].

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical Reasoning Model, July 2025. URL http://arxiv.org/abs/2506.21734. arXiv:2506.21734 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, December 2023. URL http://arxiv.org/abs/2305.10601. arXiv:2305.10601 [cs].

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning Models Know When They're Right: Probing Hidden States for Self-Verification, April 2025. URL http://arxiv.org/abs/2504.05419. arXiv:2504.05419 [cs].

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Emergence of Superposition: Unveiling the Training Dynamics of Chain of Continuous Thought, October 2025a. URL http://arxiv.org/abs/2509.23365. arXiv:2509.23365 [cs].

Rui-Jie 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, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A Survey on Latent Reasoning, July 2025b. URL http://arxiv.org/abs/2507.06203. arXiv:2507.06203 [cs].

A KV-CACHE VAE: FULL TRAINING AND EVALUATION RECIPE

A.1 TRAINING

This appendix repeats Sections 3.2 and 3.3 with every detail needed to reproduce the encoder: the frozen models and their caches, the training corpus, the architecture, the objective with all weights and schedules, the optimizer and compute, and the validity measurements. We train one KV-cache VAE per frozen model, `DeepSeek-R1-Distill-Qwen-1.5B` (***R1-1.5B***), `Qwen3-4B` and `Nemotron-Nano-4B`; the recipe is shared and each table has one column per model.

The state and its dimension. The state of Eq. 1 is the KV cache over the prefix, $S_t \in \mathbb{R}^{(|q|+t)\times D_{\mathrm{KV}}}$ with $D_{\mathrm{KV}} = 2\,L\,H_{\mathrm{kv}}\,d_{\mathrm{head}}$ fixed by the frozen model (Table 3). Each position is stored as one flat `float16` vector ordered (layer, key/value, KV head, head dimension), so a (T, D_{KV}) matrix reshapes to $(T, L, 2, H_{\mathrm{kv}}, d_{\mathrm{head}})$ and back into a per-layer `past_key_values` cache without a copy. Keys are stored after rotary position embedding, as the model holds them, and values are raw cache entries; no standardization is applied at any stage. For a hybrid architecture with recurrent layers only the attention layers write a cache, and only those are encoded and modified. Every VAE encodes at most 4096 positions, prompt included.

Table 3: **Frozen models and cache geometry.** D_{KV} is the per-token dimension of S_t and the VAE's input width.

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
HF model id	`DeepSeek-R1-Distill-Qwen-1.5B`	`Qwen3-4B`	`Llama-3.1-Nemotron-Nano-4B-v1.1`
layers L (encoded)	28 (all)	36 (all)	32 (all)
KV heads H_{kv}	2	8	8
head dim d_{head}	128	128	128
D_{KV}	14,336	73,728	65,536
reasoning delimiters	`<think>` / `</think>`	`<think>` / `</think>`	`<think>` / `</think>`
positions encoded (cap)	4096	4096	4096
frozen-model precision	bf16	bf16	bf16

Training data. The VAE trains on KV caches from the frozen model's own generations over a large math corpus (Table 4). The pipeline has four stages. **(1) Generate.** Sample one reasoning trace per problem. The problem statement is the only user message, wrapped in the model's chat template with the generation prompt appended; we save prompt+generation token ids and the indices of the first `<think>` and last `</think>` token. **(2) Replay.** Run each trace teacher-forced through the same model and write its KV cache, truncated to $[0, \texttt{think_end_idx})$: the whole prompt, the `<think>` token and every reasoning token, exclusive of `</think>` itself. Traces that never emit `</think>` are dropped here. **(3) Shard.** Consolidate into WebDataset tar shards; each sample carries the (T, D_{KV}) `float16` KV matrix, the position-aligned token ids, and `cot_len`, `think_start_idx`. **(4) Train.** A fixed block of dataset rows is held out for validation and model selection; the remaining samples under the 4096-position cap are the training data.

Table 4: **VAE training corpora.** One trace per problem. "Kept" samples closed `</think>` and fit the 4096-position table. Length quantiles are of `think_end_idx` (prompt+reasoning tokens) over the training samples.

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
source dataset	NuminaMath-CoT	DeepMath	DeepMath
HF dataset id	`AI-MO/NuminaMath-CoT`	`zwhe99/DeepMath-103K`	`zwhe99/DeepMath-103K`
generation backend	vLLM	vLLM 0.10.2	vLLM 0.10.2
decoding	greedy (T=0)	greedy (T=0)	greedy (T=0)
generation budget (tokens)	4096	4096	4096
traces generated	856,763	101,842	101,842
closed `</think>`	$\sim$56%	24.74%	8.79%
training samples kept	483,124	100,342	100,342
validation rows (held out)	first 1,500 (1,496 traces)	1,500 (1,500 traces)	1,500 (1,500 traces)
length p_{95}	3254	4095	4093

A training example is a (prefix, continuation) pair drawn freshly at each step. The prefix length is $t \sim \mathcal{U}[t_{\min}, \texttt{cot_len}]$ with $t_{\min} = \texttt{think_start_idx} + 2$, so every prefix contains the complete

prompt, the <think> token and at least one reasoning token. The continuation is the trace's next $k = \min(M, \texttt{cot_len} - t)$ tokens, where M is the horizon of Table 6. Prompt KV stays inside the prefix rather than being masked out; at evaluation the ground-truth prompt KV is spliced back over $[0, |q|)$ before anything is measured, so no metric credits or penalizes the model for the prompt. Because every continuation loss assembles a cache and runs one forward of the frozen model, which can be batched only when all samples share their prefix and continuation lengths, we sample one t per batch (the largest feasible common value) and slice every sample there. Padding is not a substitute: reconstructed keys are post-RoPE at their own positions, and a continuation rotated from a padded length would no longer be comparable to the target. Diversity in t arrives across steps rather than within a batch.

Architecture. The encoder and decoder (Dec_ψ) are 6-layer bidirectional Transformers. The encoder is fed the full KV cache S_t through a linear projection $D_{\mathrm{KV}} \to d_{\mathrm{model}}$ with fixed sinusoidal positional encoding. A learned CLS token aggregates the prefix, and linear heads on its output give the posterior

$$q_\phi(z \mid S_t) = \mathcal{N}\big(\mu_\phi(S_t),\, \mathrm{diag}\,\sigma^2_\phi(S_t)\big), \qquad \mu, \log\sigma^2 \in \mathbb{R}^{1024}. \tag{14}$$

We take the posterior mean as the latent state, $z_t := \mu_\phi(S_t)$ (Eq. 4), so the coordinate is a deterministic function of the prefix; the reparameterisation samples only in training mode. The decoder Dec_ψ projects one latent to d_{model}, broadcasts it to all cached positions, adds the same positional encoding, mirrors the encoder stack and reconstructs a full cache $\hat{S}_t$ of the same shape as S_t through a linear output layer. Table 5 lists every size. Latent dimensions above 1024 and a $12/12$-layer stack were tried on R1-1.5B without better validation performance.

Table 5: **KV-VAE architecture.** Only the input width and the resulting parameter count differ across models.

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
parameters	183.7M	305.4M	288.6M
input width D_{KV}	14,336	73,728	65,536
d_{model}	1024		
attention heads	8		
FFN width	4096		
encoder / decoder layers	6 / 6, bidirectional, pre-norm		
dropout	0.1		
positional encoding	fixed sinusoidal, 4096 positions (+1 CLS)		
readout	CLS $\to$ linear μ, $\log\sigma^2$		
latent dimension d	1024		
VAE precision	`float32` (frozen model in its own dtype)		

Training objective. The loss is a β-VAE on the cache plus a continuation term (Eq. 5),

$$\mathcal{L} = \underbrace{\|\hat{S}_t - S_t\|^2}_{\mathcal{L}_{\mathrm{recon}}} + \beta\, \underbrace{\mathrm{KL}\big(q_\phi(z \mid S_t)\,\|\,\mathcal{N}(0, I)\big)}_{\mathcal{L}_{\mathrm{KL}}} + \mathcal{L}_{\mathrm{cont}}, \tag{15}$$

where $\hat{S}_t = \mathrm{Dec}_\psi(z_t)$. Concretely, $\mathcal{L}_{\mathrm{recon}}$ is the mean squared error per position, averaged over the real positions of the prefix and over the D_{KV} coordinates, on raw cache values. The KL is taken per latent dimension, averaged over the batch, floored at a free-bits threshold τ and summed over the d dimensions,

$$\mathcal{L}_{\mathrm{KL}} = \sum_{j=1}^{d} \max\Big(\tau,\ \mathbb{E}_{\mathrm{batch}}\big[-\tfrac{1}{2}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)\big]\Big), \tag{16}$$

which prevents posterior collapse; β is annealed linearly from 0 to 1 over the first 30% of the planned steps. The first two terms address desiderata (i) and (iii). For (ii), the continuation loss has two terms (Eq. 6),

$$\mathcal{L}_{\mathrm{cont}} = \gamma_{\mathrm{tf}}\, \underbrace{\big\|F_\theta(\hat{S}_t, y_{t+1:t+k}) - S_{t+k}\big\|^2}_{\mathcal{L}_{\mathrm{tf}}} + \gamma_{\mathrm{gen}}\, \underbrace{\big\|F_\theta(\hat{S}_t, \hat{y}_{t+1:t+k'}) - S_{t+k'}\big\|^2}_{\mathcal{L}_{\mathrm{gen}}} \tag{17}$$

which differ only in where the continuation tokens come from. Both reshape the flat reconstruction into a per-layer cache, run the frozen model over the continuation tokens with it as `past_key_values`, and score the continuation KV it writes against the true continuation KV, averaged over the k positions and D_{KV} coordinates. The teacher-forced term $\mathcal{L}_{\mathrm{tf}}$ takes the trace's original k tokens $y_{t+1:t+k}$ in one parallel forward. The generative term $\mathcal{L}_{\mathrm{gen}}$ supplies only the first continuation token and lets the model autoregressively generate k' tokens $\hat{y}_{t+1:t+k'}$ greedily from $\hat{S}_t$, made differentiable with a straight-through estimator: with $p = \mathrm{softmax}(\ell)$ and $h = \mathrm{onehot}(\arg\max \ell)$ the surrogate $\tilde{y} = h - \mathrm{sg}(p) + p$ has the one-hot as forward value and the softmax's gradient, and the next input embedding is $\tilde{y}E$. This runs k' sequential frozen-model forwards retained in the autograd graph, which is what bounds the horizon. Table 6 gives every weight.

Table 6: **Objective hyperparameters.** Symbols as in Eqs. 5 and 6. A single horizon flag M sets both k and k'.

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
reconstruction weight	1	1	1
β schedule	linear $0 \to 1$, first 30% of steps	linear $0 \to 1$, first 30% of steps	linear $0 \to 1$, first 30% of steps
free bits τ (nats/dim)	0.5	0.5	0.5
continuation scored in	KV space	KV space	KV space
γ_{tf}	1.0	0.1	0.1
k (teacher-forced horizon)	8	32	32
γ_{gen}	0.1	0	0
k' (generative horizon)	8	not used	not used
shared prefix length per batch	yes	yes	yes

We tried different variants of the continuation loss before arriving on the above terms. They differ in where the mismatch is scored. *(KV-space)* MSE between the continuation KV produced from the reconstructed prefix and the ground-truth continuation KV, as above; the tightest signal and the cheapest. *(Latent-space)* MSE in the encoder's own latent between the encoding of [true prefix ‖ true continuation KV] and of [true prefix ‖ predicted continuation KV]; the encoder is frozen for this term so the distance is a fixed yardstick that cannot be shortened by collapsing the metric. *(Token-space)* cross-entropy of the frozen model's next-token distributions under the reconstructed prefix against the true continuation tokens, which never sees the continuation KV. Each can be combined with teacher-forced or generated continuation tokens.

Table 7: **Optimizer and run length.** "Planned" steps correspond to the configured epoch budget; runs were stopped by allocation and the best-validation checkpoint is used.

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
optimizer	AdamW	AdamW	AdamW
learning rate	10^{-4}	10^{-4}	10^{-4}
weight decay	10^{-2}	10^{-2}	10^{-2}
gradient clipping (norm)	1.0	1.0	1.0
LR schedule	cosine to $0.1\times$	cosine to $0.1\times$	cosine to $0.1\times$
batch size per rank	4	4 ($\times 8$ accumulation)	4 ($\times 8$ accumulation)
GPUs (DDP ranks)	64 (8 nodes $\times$ 8 GCDs)	4 (4×H200)	4 (4×H200)
effective batch	256	128	128
planned steps (epochs)	56,610 (30)	783 (1)	783 (1)
steps completed (epochs)	19,800 (10.5)	783 (1)	783 (1)
selected checkpoint	step 16,983 (epoch 9)	step 783 (epoch 1)	step 783 (epoch 1)

A.2 VAE VALIDITY

We use z_t as a shared coordinate for the full prefix state. Before analyzing dynamics in the VAE's coordinate system, we confirm the trained VAE performance against baselines. We refer to a resampled baseline where tokens y' are resampled in place of the trace's original tokens, so that $S'_t = F_\theta(S_0, y'_{1:t})$ is an independent trace of the same problem and $S'_{t+k} = F_\theta(S_t, y'_{t+1:t+k})$ is a resampled continuation of the same prefix. All measurements use traces held out from VAE training, at sampling temperature 0.6, and are reported per model in Table 8.

Reconstruction. The decoded cache should be closer to its source than a different trace of the same problem is at the same depth, $\|\hat{S}_t - S_t\| \ll \|S'_t - S_t\|$. We report the relative error $\|\hat{S}_t - S_t\|/\|S_t\|$ at fixed depths t from 128 to 4096; on R1-1.5B it is 0.08 and flat from t=128 to 3072 even though the training traces are short (Table 4), so the deep rows of the landscapes measure dynamics, not encoder failure.

Continuation. Running the frozen model from the reconstructed cache $\hat{S}_t$ on the trace's original next k tokens should land closer to the true continuation than resampling k different tokens from the true cache S_t, $\|F_\theta(\hat{S}_t, y_{t+1:t+k}) - S_{t+k}\| \ll \|F_\theta(S_t, y'_{t+1:t+k}) - S_{t+k}\|$. We score the same comparison in KV space and in token space as the per-token KL between the next-token distributions under $\hat{S}_t$ and under S_t.

Continuation accuracy. Generating from the decoded cache should solve the problem about as often as generating from the true one, $P_\theta(\text{correct} \mid F_\theta(\hat{S}_t, y_{t:T})) \approx P_\theta(\text{correct} \mid S_T)$, measured from decoded prefixes at $t = 25\%, 50\%, 75\%$ of a trace. The reconstructed cache reproduces the original trace's exact *specific* answer only at a similar rate to a resample, so all observational results use the encoder alone and the interventions of Section 3.5 add $\text{Dec}(z + \delta v) - \text{Dec}(z)$ to the true cache instead of replacing it.

Closure. The dynamics on z are not closed by construction, since z_{t+1} is a fresh encoding of S_{t+1} and not a function of z_t. To test Markovianness, we take two states S_a, S_b from different traces of one problem with $\|z_a - z_b\| \le r$ (See Figure **??**). Then generate o additional tokens from each state and measure the separation of the encoded results, $D(o) = \|\text{Enc}_\phi(F_\theta(S_a, y^a_{1:o})) - \text{Enc}_\phi(F_\theta(S_b, y^b_{1:o}))\|$. Two references bound it: the floor $D_{\text{floor}}(o)$, two resamples of one identical prefix, which is the separation sampling noise alone produces, and the ceiling $D_{\text{ceil}}(o)$, two unrelated states at matched depth. The closure residual

$$\rho(o) = \frac{D(o) - D_{\text{floor}}(o)}{D_{\text{ceil}}(o) - D_{\text{floor}}(o)} \tag{18}$$

is 0 if z is a state and 1 if it carries no information about the future. At $o = 512$ we find $\rho = 0.20$ $[0.17, 0.22]$ over 478 pairs, unchanged across sampling temperatures 0.6, 1.0 and 1.5, and monotone in the initial separation: 0.83 $[0.75, 0.92]$ for pairs 3.3 apart down to 0.14 $[0.06, 0.24]$ for pairs 0.8 apart. Nearby in z means nearly the same future, up to sampling noise, and we treat z as an approximate state on that basis.

Table 8: **VAE validity per model.** Reconstruction is reported at the listed reasoning depths, with the mean in parentheses. Brackets are 95% intervals. KL and continuation accuracy are reported at $t = 25/50/75\%$.

Model	reconstruction depths	reconstruction error	mean
R1-1.5B	128/1024/3072/4096	0.076/0.077/0.078/0.090	0.080
Qwen3-4B	64/256/1024/2048	0.099/0.097/0.096/0.096	0.097
Nemotron-Nano-4B	64/256/1024/2048	0.106/0.105/0.103/0.103	0.104

	R1-1.5B	Qwen3-4B	Nemotron-Nano-4B
per-token KL (nats), $t = 25/50/75\%$	0.30/0.53/0.95	0.33/0.58/1.00	0.34/0.61/1.00
continuation accuracy from $\hat{S}_t$, $t = 25/50/75\%$	0.60/0.58/0.56	0.65/0.63/0.60	0.67/0.65/0.63
accuracy from the true prefix	0.64	0.69	0.72
closure $\rho(512)$	0.20 [0.17, 0.22]	0.23 [0.20, 0.26]	0.25 [0.22, 0.28]

B Analysis details

DeepSeek-R1-Distill-Qwen-1.5B Our primary analysis uses temperature $T = 0.6$. We perform all analyses on MATH-500. The error field is evaluated at 28 reasoning depths using forced-close answers with a 1,024-token answer budget and strict boxed-answer grading. For gradient estimation, we construct per-problem PCA bases; for latent interventions, we use the top eight global principal

components. Unless otherwise specified, we report statistics per problem and use sign-consistency filtering, shuffled-label nulls, and bootstrap confidence intervals.

Qwen3-4B. We repeat the analyses of the Results section on `Qwen/Qwen3-4B` (thinking mode), with a KV-cache VAE trained on DeepMath traces of that model. Latents are stored every $\Delta = 4$ reasoning tokens; the error field is probed on 16 traces per problem at 28 depths with 16 forced-close answers ($T = 0.6$, answer cap 2048), giving $\sim$420 probed states per problem. That is roughly a seventh of the density behind Figure 4, which matters for the local gradient estimator below.

Nemotron-Nano-4B. We apply the same pipeline to `nvidia/Llama-3.1-Nemotron-Nano-4B-v1.1`, with its own VAE and its own probed field (same protocol as Appendix B). The intervention algorithms were not rerun on this model.

C CAUSAL INTERVENTION ALGORITHMS AND RESULTS

Active repulsion samples B traces one at a time. During the first quarter of each trace's budget, if a state S_t lies closer to a visited state than the lower quartile of the pool's pairwise distances, it is displaced away from its five nearest visited states. The displacement decays linearly from $\delta = 16$ to zero over that quarter. Algorithm 1 gives the full procedure. We find that this algorithm can slightly degrade performance in current models. We leave to future work whether reinforcement learning alongside this method, or other forms of latent regularization during RL, can improve sampling performance.

We use $\mathrm{normalize}_0(v) = v/\|v\|$ for $v \neq 0$ and $\mathrm{normalize}_0(0) = 0$. Pairwise thresholds use distinct pool entries; repulsion starts only once at least two states are available.

Algorithm 1 Repulsion anneal: breadth as coverage of the latent manifold. Settings used here: B=8, T=16,384, c=512, δ=16, a=0.25, q=0.25, PCA basis V = top k=8 global components, $L_{\max}$=4096. Here T denotes the token budget.

Require: prompt x; breadth B; budget T; chunk c; displacement δ; anneal cutoff a; threshold quantile q; PCA basis V; maximum prefix length $L_{\max}$

$\mathcal{V} \leftarrow \emptyset$ ▷ every latent seen, pooled across *all* traces
for $i = 1 \dots B$ **do**
  $u \leftarrow x$
  while $\mathrm{reason}(u) < T$ **and** u has not emitted EOS **do**
    $c_t \leftarrow \min(c, T - \mathrm{reason}(u))$; $L \leftarrow |u|$
    if $L > L_{\max}$ **then** ▷ VAE position buffer is 4096; cannot encode
      $u \leftarrow u \parallel \textsc{Generate}(u, c_t)$; **continue**
    end if
    $z \leftarrow \mathrm{Enc}\big(\mathrm{KV}(u)_{1:L}\big)$ ▷ latent of the whole prefix
    $\rho \leftarrow \mathrm{reason}(u)/T$ ▷ progress through the budget
    hit $\leftarrow$ FALSE
    if $\rho < a$ **and** $|\mathcal{V}| \geq 2$ **then**
      $d \leftarrow \min_{w \in \mathcal{V}} \|z - w\|$
      $\tau \leftarrow q$-quantile of the pairwise distances within $\mathcal{V}$ ▷ adaptive, not absolute
      hit $\leftarrow (d < \tau)$ ▷ this region is already occupied
    end if
    if *hit* **then**
      $N \leftarrow$ the $\min(5, |\mathcal{V}|)$ nearest elements of $\mathcal{V}$ to z
      $v \leftarrow \frac{1}{|N|} \sum_{w \in N} (z - w)$ ▷ "far": away from the local crowd
      $v \leftarrow \mathrm{normalize}_0\big(\mathrm{proj}_V(v)\big)$ ▷ project on-manifold, else the trace stops terminating
      $\delta_t \leftarrow \delta \max(0, 1 - \rho/a)$
      $\Delta\mathrm{KV} \leftarrow \mathrm{Dec}(z + \delta_t v, L) - \mathrm{Dec}(z, L)$
      $u \leftarrow u \parallel \textsc{Generate}\big(\mathrm{KV}(u) + \Delta\mathrm{KV},\, c_t\big)$ ▷ added to the *true* cache
    else
      $u \leftarrow u \parallel \textsc{Generate}(u, c_t)$
    end if
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{z\}$
  end while
  yield u
end for

Qwen3-4B: results on three benchmarks. We run multiple seeds and active repulsion on MATH-500 (the same 32 random problems as Table 9), AMC23 (all 40 problems) and GSM8K (100 problems drawn with a fixed seed), with $B = 8$, $T = 0.6$ and a budget of 16,384 tokens. Multiple seeds use $\delta = 8$ in the top-8 PCs of a steering basis fit on held-out DeepMath prefixes, so no benchmark outcome chooses a direction. Repulsion uses $\delta = 16$ tapered linearly to 0 over the first 25% of the budget, away from the 5 nearest visited states (Algorithm 2); it runs on the 32-problem subsets only. Past the anneal window no chunk is perturbed, so those tokens are generated with the paged sampler, which changes only the random stream. A branch that stops without an answer counts as incorrect.

Algorithm 2 Repulsion anneal, simplified. Algorithm 1 gives the full procedure, settings, and implementation constraints.

$\mathcal{V} \leftarrow \emptyset$ ▷ every latent seen, pooled across *all* traces
for $i = 1 \dots B$ **do**
  $u \leftarrow$ prompt
  while u is still generating **do**
    $z \leftarrow \mathrm{Enc}\big(\mathrm{KV}(u)\big)$ ▷ latent of the prefix
    if u is early in its budget **and** z is unusually close to $\mathcal{V}$ **then**
      $v \leftarrow$ on-manifold direction away from the nearest members of $\mathcal{V}$
      $\delta_t \leftarrow \delta \max(0, 1 - \mathrm{reason}(u)/(aT))$
      extend u by up to c tokens from $\mathrm{KV}(u) + \big[\mathrm{Dec}(z + \delta_t v) - \mathrm{Dec}(z)\big]$
    else
      extend u by up to c tokens as usual
    end if
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{z\}$
  end while
end for

Table 9: **Latent restart delta sweep.** Latent restart vs. temperature sampling at $B = 8$, 32 random MATH-500 problems, matched token budget. "distinct" is the number of distinct answers among the eight restarts.

arm	pass@8	distinct	mean	majority	tokens
$\tau = 0.6$	0.733	2.13	**0.636**	0.685	**45.8k**
$\tau = 1.0$	0.718	2.32	0.605	0.666	48.2k
$\delta = 0$ (control)	0.719	2.06	0.629	**0.688**	70.5k
$\delta = 4$	0.750	2.38	0.602	0.656	75.0k
$\delta = 8$	**0.812**	3.00	0.535	**0.688**	88.1k
$\delta = 16$	0.750	**4.12**	0.375	0.531	102.6k

Every arm is scored on B-subsets of its own branches, so pass@B always compares groups of equal size, and differences are paired bootstraps over problems.

Repulsion annealing is harder to implement efficiently than latent restarts because it steers the model repeatedly during decoding; its gains are also less consistent.

Table 10: **Qwen3-4B repulsion at $B = 8$, depth 8,192.** Δ is the paired difference from $T = 0.6$ with a 95% bootstrap interval; "breadth" is pass@8 − pass@1.

set	arm	n	pass@1	pass@8	distinct	Δ breadth
MATH-500	$T = 0.6$	32	0.869	0.943	1.43	—
	seeds $\delta = 8$	32	0.867	0.957	1.50	+0.015 [−0.021, +0.050]
	repulsion $\delta = 16$	32	0.716	0.931	1.69	+0.116 [+0.032, +0.198]
AMC23	$T = 0.6$	40	0.832	0.943	1.51	—
	seeds $\delta = 8$	40	0.805	0.948	1.45	+0.033 [+0.000, +0.066]
	repulsion $\delta = 16$	32	0.585	0.939	2.06	+0.214 [+0.130, +0.302]
GSM8K	$T = 0.6$	100	0.929	0.963	1.12	—
	seeds $\delta = 8$	100	0.896	0.980	1.22	+0.050 [+0.031, +0.072]
	repulsion $\delta = 16$	32	0.841	0.976	1.13	+0.094 [+0.049, +0.147]

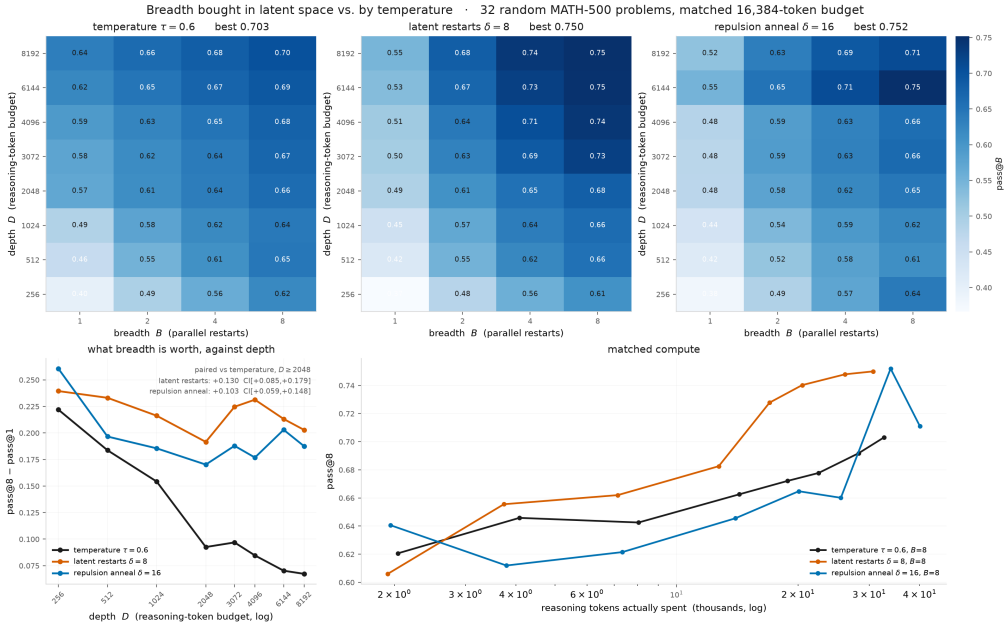

Figure 5: **Depth × breadth for two inference-time algorithms. Top.** Pass@B improvement over standard temperature sampling (red cells are worse than baseline and green are better). Latent restarts start lower at B=1 and end higher at B=8. **Bottom left.** What breadth is worth, pass@8 − pass@1, against depth—the gain from temperature sampling falls by a factor of three as the traces lengthen while both latent arms stay flat. **Bottom right.** The same arms against tokens actually spent.

D DIRECTION: THE POLICY SETS THE DIRECTION OF THE FLOW

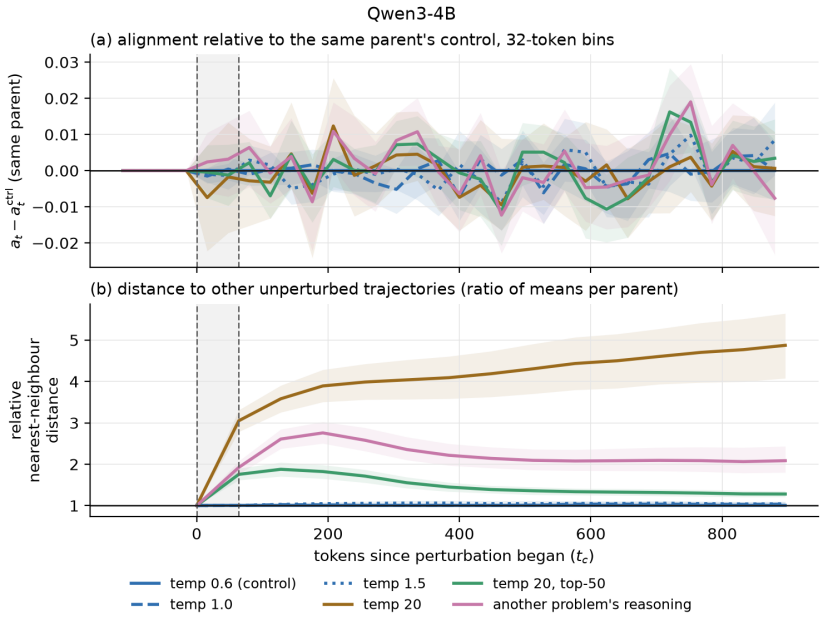

Figure 6: **Qwen3-4B: token injection** Here we reproduce the results in Figure D for the Qwen3-4B model. (a) the local gradients are too noisy to show a confirm the same results as found in the main figure. We believe this is due to too large granulartiy taken between steps. (b) The relative rankings for the arms hold as in the original plot.

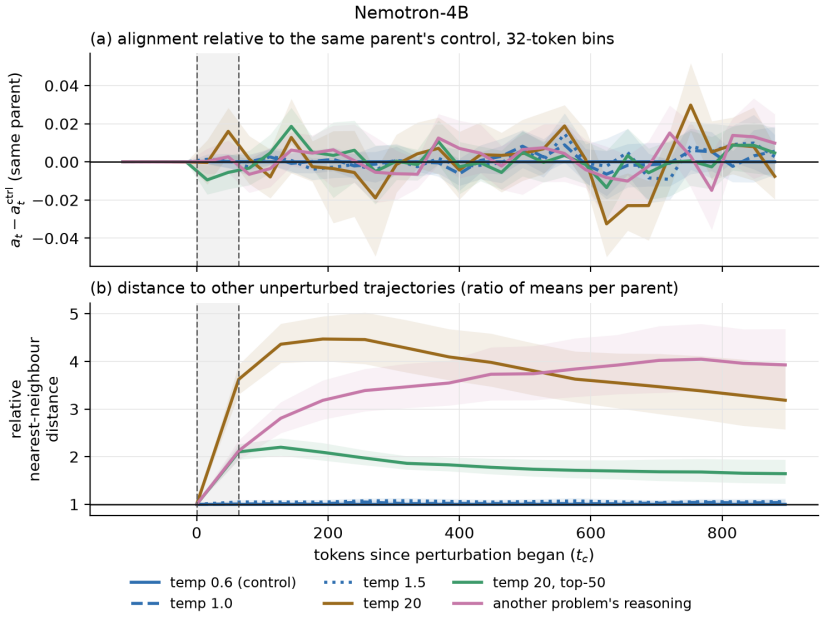

Figure 7: **Nemotron-Nano-4B: token injection** Here we reproduce the results in Figure D for the Nemotron-Nano-4B model. (a) the local gradients are too noisy to show a confirm the same results as found in the main figure. We believe this is due to too large granulartiy taken between steps. (b) The relative rankings for the arms hold as in the original plot.

E TOKEN SCAN: EXPLORATION AND EXPLOITATION

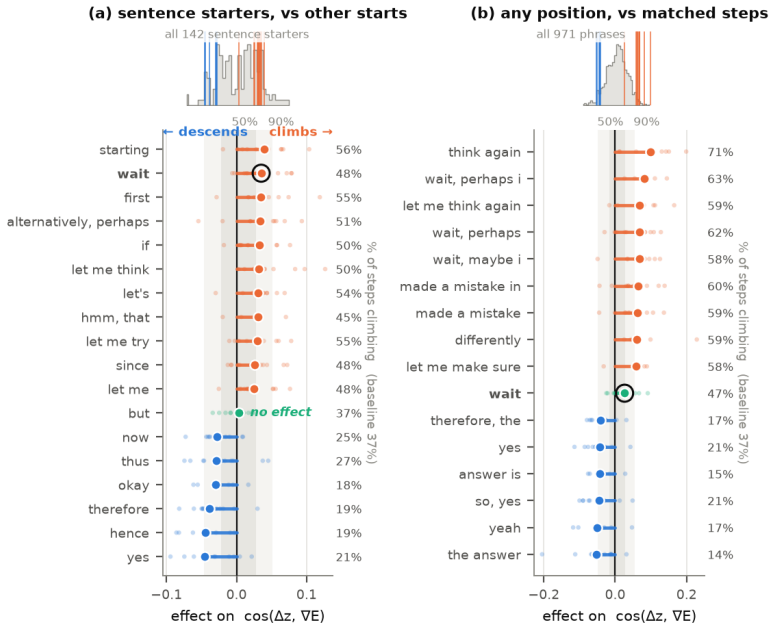

Figure 8: **Token scan: exploration and exploitation.** Token and phrase associations with latent step direction.

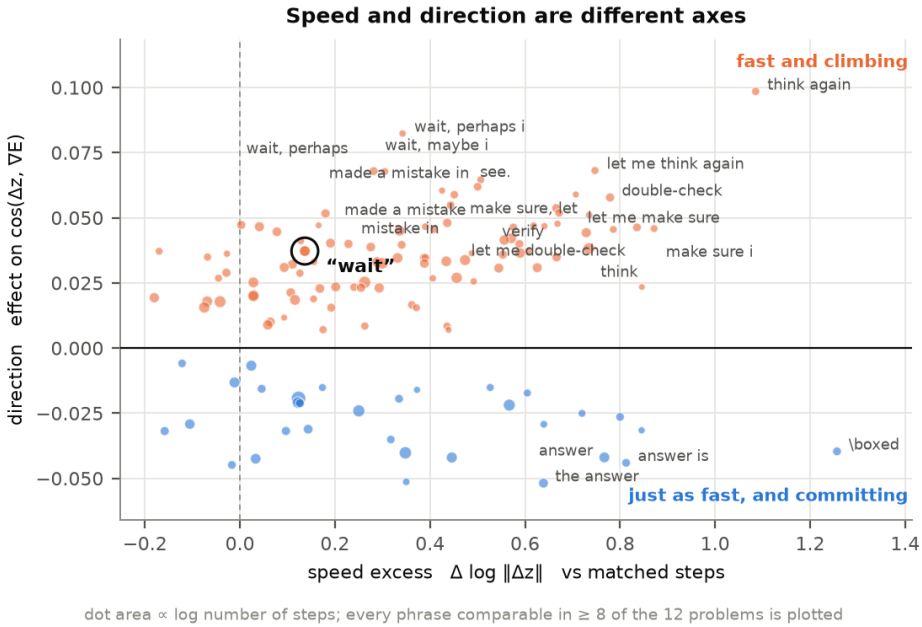

Figure 9: **Token-scan scatter plot.** Complementary visualization of the token-scan analysis in Figure 8.

Qwen3-4B.

Controlling exploration and exploitation. Qwen3-4B additionally supports the main phrase-level pattern: checks and replanning phrases tend to accompany ascent, while commitment phrases accompany descent, but the effects are sligtly smaller (Figure 10). Nemotron-Nano-4B incurs addi-

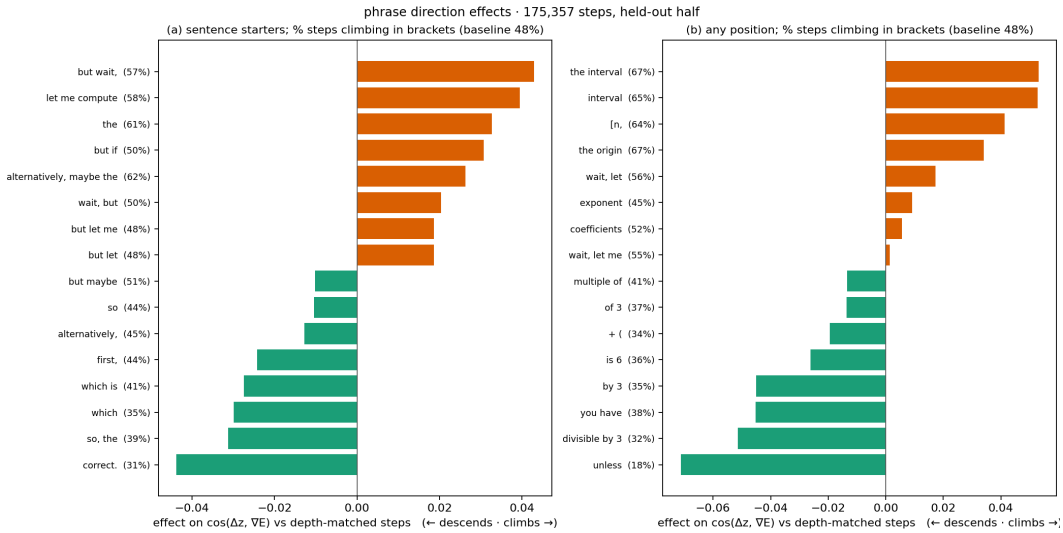

Figure 10: **Qwen3-4B: phrase effects on step direction.** Every stride-4 step of the probed traces (175,357 steps, 192 traces, depth $\geq$ 128) scored against the local gradient at its midpoint. Phrases are selected on half the traces and effects estimated on the held-out half, against steps matched on problem and depth bin; brackets give the share of steps climbing.

tional noise, which is filtered out by focusing on sentence starters, where commitment phrases still show a descending association (Figure 11).

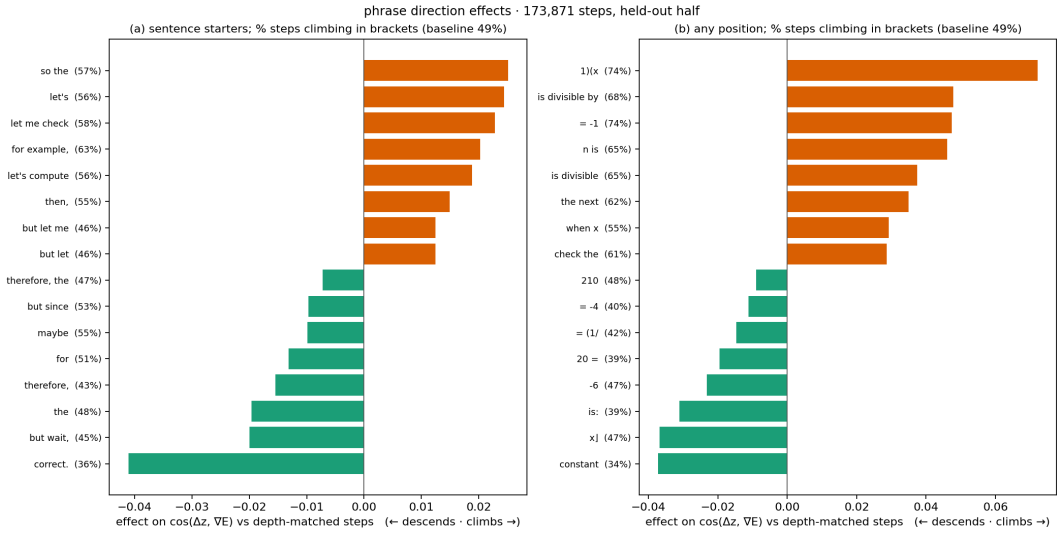

Figure 11: **Nemotron-Nano-4B: phrase effects on step direction** (173,871 steps), as Figure 10.

Nemotron-Nano-4B. The phrase scan is essentially null on this model (Figure 11): against a 49% baseline, only *correct.* (36%) and *therefore,* (43%) depart clearly, both descending.

F SPEED: THE FLOW SLOWS WITH DEPTH, NOT WITH THE LANDSCAPE

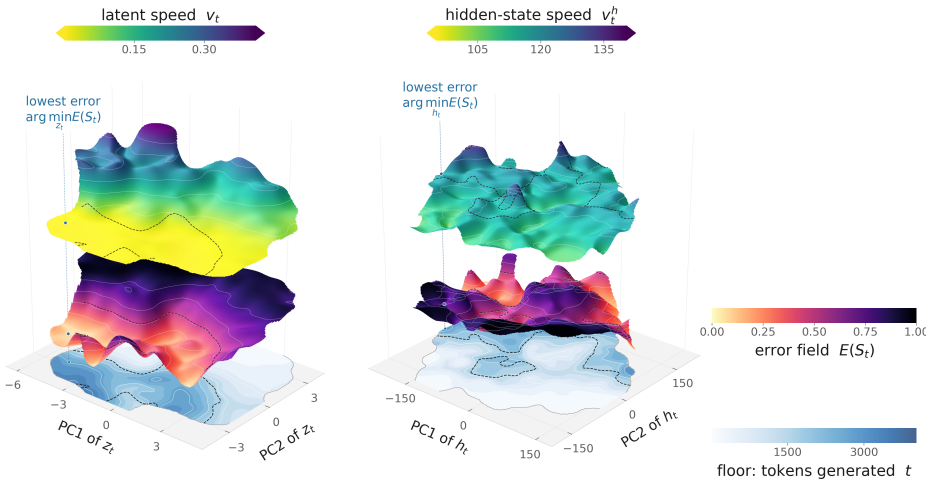

Figure 12: **Latent speed from the KV-VAE encoder compared with speed computed from hidden states.** In this example, the hidden-state representation matches the error manifold less closely than the KV-VAE representation.

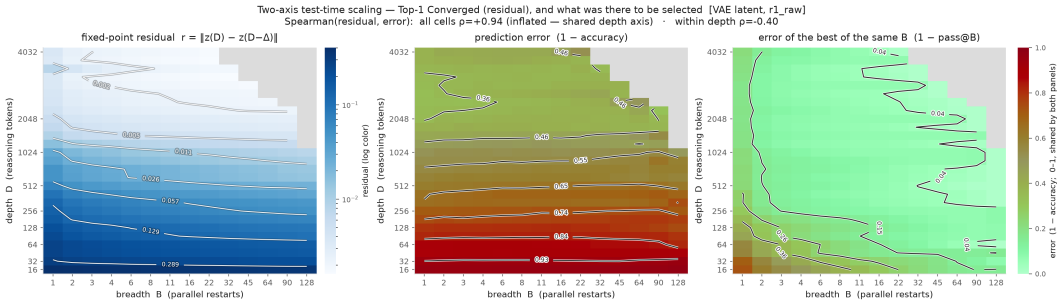

Figure 13: **Two-axis test-time scaling.** Residual speed and best-of-B error, $1 - \text{pass@}B$, across reasoning depth and sampling breadth. Level sets compare how the two quantities vary with depth and breadth.

Qwen3-4B. Both models confirm the main speed result: latent speed resembles the error field more closely than hidden-state speed. (Figures 15 and 17). Depth predicts held-out speed with $R^2 = 0.96$ for Qwen3-4B and 0.98 for Nemotron-Nano-4B. Increasing breadth improves success while leaving speed nearly unchanged, confirming that slower latent motion is not itself a reliable measure of progress toward correctness (Figures 16 and 18).

Nemotron-Nano-4B. Both speed results reproduce (Figures 17 and 18). Over the 13 informative problems the latent speed field follows the error field (median $+0.42$) and depth (-0.63) while the hidden-state residual carries nothing (-0.06), with latent speed the stronger signal on all 13; depth alone predicts held-out speed with $R^2 = 0.98$, and speed varies by at most 8.5% across breadths at any depth while $1 - \text{pass@}B$ falls with B.

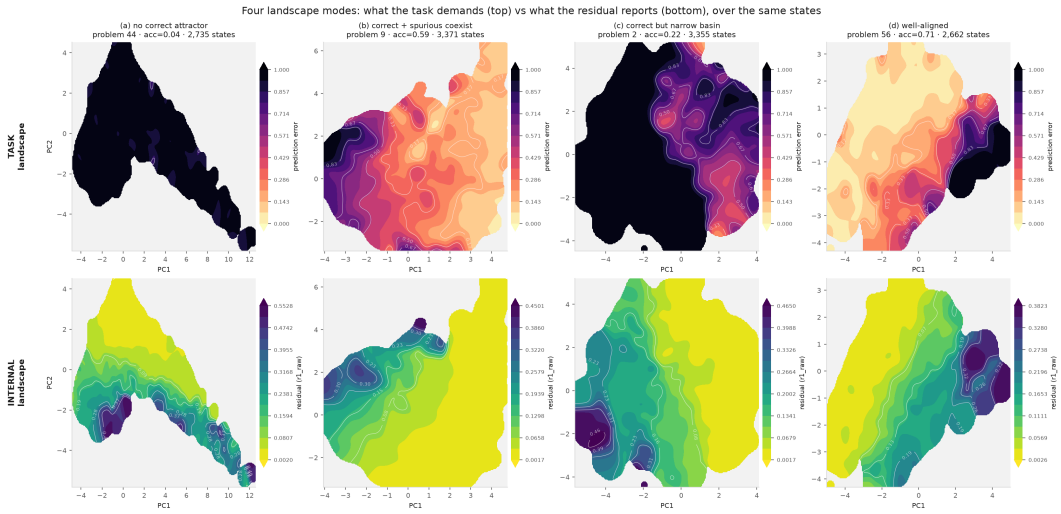

Figure 14: **Four landscape modes: task landscape (top) vs internal landscape (bottom), over the same states.** Column j is one problem. The equilibrium claim is that the two rows should look alike. Note column (a): the task landscape is featureless—nothing is ever correct anywhere—yet the residual reports a confident, well-organized gradient. It is reporting depth, and depth is meaningless there.

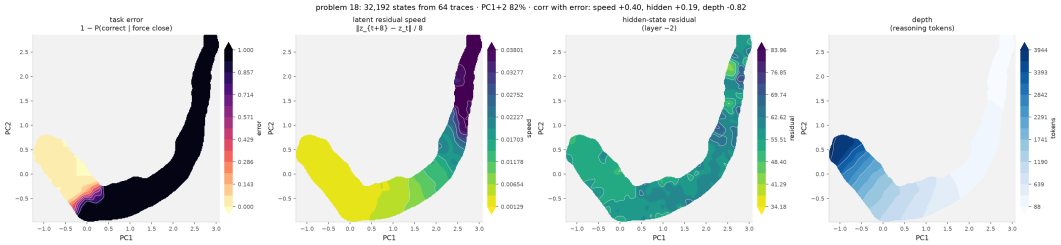

Figure 15: **Qwen3-4B: four fields for one MATH-500 problem.** 32,192 states from 64 traces: every 8 reasoning tokens the error is measured by forcing the answer (four resamples) and the latent and the layer -2 hidden state are taken at the same prefix; PC1+2 explain 82%. Smoothed with the kernel of Figure 4. Correlation with the error field: latent residual speed $+0.40$, hidden-state residual $+0.19$, depth -0.82.

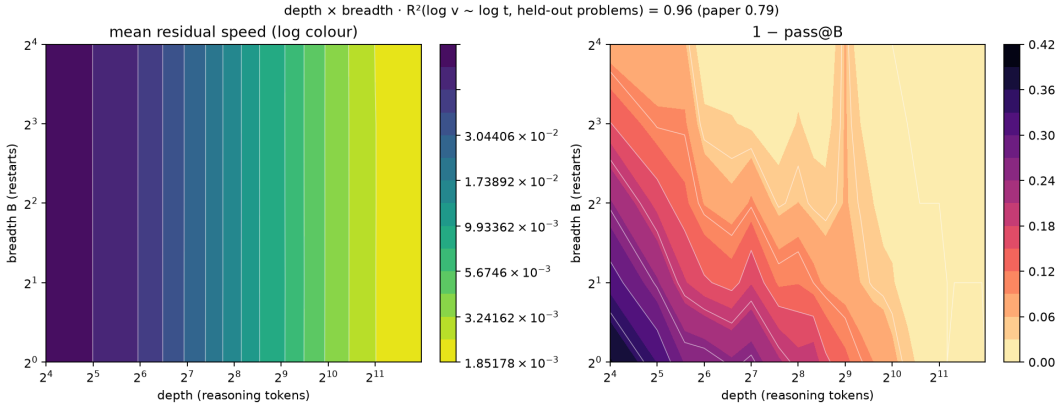

Figure 16: **Qwen3-4B: two-axis test-time scaling.** Left: mean residual speed (log colour scale). Right: $1 - \text{pass@}B$. Breadths above 16 are omitted (16 traces per problem).

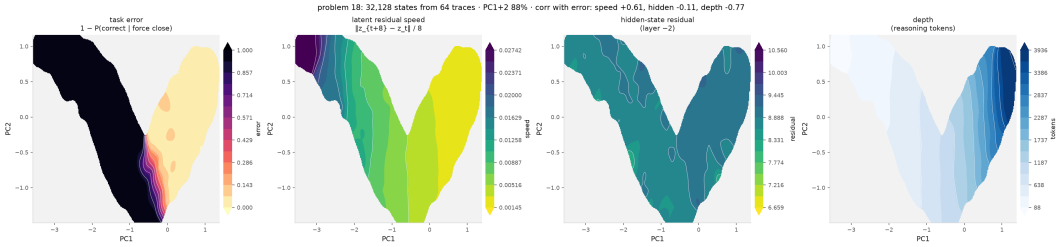

Figure 17: **Nemotron-Nano-4B: four fields for one problem**, measured as Figure 15 (32,128 states from 64 traces). Correlation with error: latent speed $+0.61$, hidden-state residual -0.11, depth -0.77.

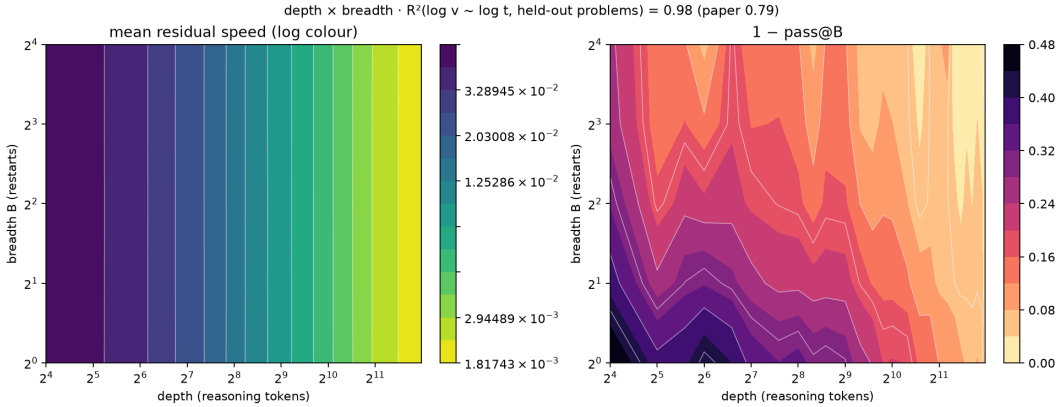

Figure 18: **Nemotron-Nano-4B: two-axis test-time scaling**, as Figure 16.

G RL TRAINING SETUP

Setup. We train Qwen3-4B with GRPO on 2,048 problems from the MATH training split (no overlap with the MATH-500 problems below), in a single-turn setting: the question is the only user message, and the reward is 1 if the last `\boxed{}` after `</think>` matches the answer. Batches have 16 problems $\times$ 32 samples, the learning rate is a constant 10^{-6}, and 128 steps make one pass over the set. Over training the per-sample reward rises from 0.81 (steps 0–15) to 0.90 and levels off around step 50, truncation at the 8,192-token cap falls from 15.7% to 6.1%, and the median response shortens from 3,683 to 1,855 tokens and keeps shortening after the reward plateaus. We study checkpoints at steps 31, 95 and 127. For each, the base VAE is fine-tuned (warm start, 180 steps) on that checkpoint's own rollouts from the preceding 16 steps, with the checkpoint as the frozen model. The fine-tunes reach a validation ELBO of within 0.1 of the VAE for the base model. VAE on base traces, indicating similar reported validation objectives across checkpoints.

Error fields. Each checkpoint generates 64 traces ($T = 0.6$, 4,096-token cap) on the same frozen 32 random MATH-500 problems. Every 8 reasoning tokens we force `</think>` and the prefix *Then my answer is* `\boxed{`, sample four completions, and grade them; the latent at the same prefix comes from that checkpoint's VAE, so the two fields join per state without interpolation. Per problem the states are projected onto their own top two PCs and E is smoothed with the kernel of Figure 4. A basin is a connected region in the bottom 20% of the smoothed field (at least 20 grid cells), and fields with range below 0.15 are treated as flat.