REINFORCING LLM LATENT REASONING WITH CONTEXTUAL LOOPS THROUGH ON-POLICY FLOW-MAP SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent reasoning creates a continuous workspace for solving a problem and a context for generating its answer. We conceptually connect recurrent Transformers, continuous diffusion, and finite-time transport and introduce contextual loops, where the same model only recursively refine a latent workspace before performing standard autoregressive decoding. Latent reasoning thereby becomes a source of privileged teacher context whose construction can be controlled through refinement depth and latent space design. We thus introduce joint reward-weighted on-policy self-distillation (OPSD) over the latent and autoregressive states of this process. At student-visited latent states, composed flow maps teach a single update to reproduce several finer refinements. At student-generated texts, the same model uses a more extensively refined latent to supervise its next-token predictions. Verifier rewards weight both objectives, linking the compression of latent computation to the explicit policy that interprets its result. Our instance, LaDi-OPSD, jointly optimizes the two OPSD objectives together with a hierarchical reinforcement objective. The resulting training scheme supports several latent inference loop budgets within one shared model. Our theoretical analysis characterizes reward-weighted supervision and the conditions for transferring teacher quality to a smaller computation budget. Across several coding and mathematical reasoning benchmarks, LaDi-OPSD outperforms other diffusion language models and latent reasoning baselines, while preserving both generation diversity and few-step generation capability.

1 INTRODUCTION

"What can be said at all can be said clearly, and whereof one cannot speak thereof one must be silent."
— Ludwig Josef Johann Wittgenstein, *Tractatus Logico-Philosophicus*

Large language models (LLMs) have demonstrated strong reasoning capabilities across challenging mathematics, coding and general-purpose tasks (Wei et al., 2022; Shao et al., 2024; Yang et al., 2025). Reasoning can take many forms: chain-of-thought (CoT) externalizes intermediate decisions as explicit text (Wei et al., 2022), whereas latent reasoning allocates computation to continuous representations that guide an answer. Within latent reasoning, recurrent approaches repeatedly reuse a shared Transformer computation: looped Transformers (LT) refine hidden states through recurrent depth (Saunshi et al., 2025; Geiping et al., 2025), while Coconut feeds each continuous thought back as the input embedding of the next reasoning step (Hao et al., 2025). Latent diffusion models instead progressively transform noise into reasoning representations, and could be either decoded directly (Gong et al., 2023) or combined with explicit discrete texts (Kang et al., 2026b; Zhou et al., 2026; Zheng et al., 2026; Pynadath et al., 2025).

These latent reasoning approaches admit a common procedure of iterative computation over continuous states: a shared Transformer repeatedly maps an evolving reasoning state to its next refinement, with the number of iterations determining the effective computation depth. Looped Transformers realize this via vanilla recurrent updates, while continuous diffusion parametrize the generation process as denoising with an SDE or ODE. While being at least the same expressive as LT under mild conditions, continuous diffusion can be trained more efficiently and supports more flexible inference, since its number of function evaluations (NFEs) provides a natural control over the refinement depth (Zhou

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.

et al., 2026). Moreover, a finite-time flow map learns the effect of an entire interval (Boffi et al., 2025), allowing one evaluation to approximate several finer updates.

In this paper, we aim to retain the benefits of latent reasoning while using as few loops as possible. We study a particular instance that we call *contextual loops*, in which a shared Transformer reads the problem and recurrently updates only a latent workspace. Once the latent loop terminates, the model conditions on the resulting workspace and generates the response through standard autoregressive decoding, without looping over the output tokens. Our central contribution is *joint reward-weighted on-policy self-distillation (OPSD) over latent and autoregressive states*. A student rollout constructs latent workspaces, samples responses, and obtains verifier rewards. At student-visited latent states, compositions of shorter updates supervise flow maps spanning longer intervals, teaching each evaluation to approximate more refinement. At student-generated prefixes, the same model constructs a paired teacher workspace from the same prompt and initial noise but with a larger latent-computation budget; its token distributions then supervise the decoder conditioned on the student workspace. Verifier rewards weight both distillation objectives, which are jointly optimized with hierarchical reinforcement learning over the latent trajectory and generated response. Sharing rollouts, rewards, and backbone parameters couples workspace compression to the policy that interprets its output.

Our key insight is that latent computation provides not only an inference-time resource but also a *supervision dimension*. Finer updates supply targets for coarser flow maps, while additional latent refinement creates privileged context for contextual OPSD (Zhao et al., 2026). Thus, the same refinement axis supports two complementary goals: spending more loops for test-time scaling and transferring their benefit into a smaller student budget for efficient inference. Joint on-policy training keeps both transfers aligned with the evolving state distribution and with downstream task reward. Our instance, LaDi-OPSD, adopts the latent representation and initialization of LaDiR (Kang et al., 2026b) and combines the two OPSD objectives with LaDi-RL's hierarchical reinforcement objective (Kang et al., 2026a). The resulting post-training procedure adapts both workspace construction and autoregressive readout across multiple loop budgets. We also provide theoretical analysis on reward-weighted supervision and the conditions under which refined teacher context improves a smaller-budget student. In both coding and math experiments, LaDi-OPSD significantly outperforms discrete diffusion models including LLaDA (Nie et al., 2025) and DREAM (Ye et al., 2025), as well as latent reasoning baselines LaDiR and LaDi-RL. Remarkably, the margins in pass@k evaluations remain significant for large k, and the performance does not drop with even only one latent generation step, suggesting that our method enables powerful yet efficient latent reasoning with as few as only one loop while preserving the generation diversity.

2 Related Work

Latent reasoning and recurrent computation. Universal Transformers introduce shared recurrent self-attention (Dehghani et al., 2019), while recurrent-depth language models and Ouro scale inference computation through latent iteration (Geiping et al., 2025; Zhu et al., 2025). Coconut represents successive thoughts as continuous hidden states (Hao et al., 2025); LOTUS combines looped computation with parallel supervision over latent reasoning blocks (Fan et al., 2026). LaDiR constructs reasoning representations through latent diffusion (Kang et al., 2026b), and LaDi-RL reinforces their downstream utility through hierarchical latent–text rollouts (Kang et al., 2026a). Our contextual-loop formulation connects this iterative computation to continuous transport over a persistent workspace that conditions autoregressive decoding.

Flow maps and few-step computation. Progressive distillation compresses denoising transitions (Salimans & Ho, 2022), and consistency models learn trajectory-to-endpoint mappings (Song et al., 2023). Shortcut models condition updates on their duration and supervise them through composition (Frans et al., 2025); MeanFlow models interval-average velocity (Geng et al., 2025). Align Your Flow develops continuous-time objectives for transport between noise levels (Sabour et al., 2025), and Boffi et al. (2025) organize flow-map self-distillation through differential and composition identities. We apply composed-map supervision to student-visited latent states, with verifier rewards allocating weight to workspaces that support successful reasoning.

On-policy and contextual self-distillation. Generalized Knowledge Distillation supervises student-generated sequences and combines distillation with reinforcement learning (Agarwal et al., 2024).

Contextual self-distillation supplies a shared model with additional teacher information: verified reasoning traces in OPSD (Zhao et al., 2026), later masked-denoising states in dOPSD (Dat et al., 2026), and learned experience-conditioned latent tokens in LOPD (Zhang et al., 2026). Our contribution is joint reward-weighted on-policy self-distillation of latent flow-map transitions and context-conditioned token predictions. Appendix A further develops the connections to representation learning, adaptive computation, hybrid diffusion, and reward-weighted policy fitting.

3 METHOD

We present a conceptual unification of loop models with iterative function calls, and introduce contextual loop as one instance (Section 3.1). To post-train them, we propose joint reward-weighted on-policy self-distillation over the latent and autoregressive states of a contextual-loop model (Section 3.2). Composed flow maps supervise the construction of a reasoning workspace, and a refined latent context supervises the explicit policy that uses it. Furthermore, shared rollouts and verifier rewards couple these objectives with reinforcement learning, training the model to solve problems across latent computation budgets (Section 3.3).

3.1 CONTEXTUAL LOOPS FOR LATENT AND EXPLICIT REASONING

A contextual latent workspace. Let q denote a problem or a prompt, $z \in \mathbb{R}^{L\times d}$ a latent workspace that can be iteratively refined, and $y = (y_1, \dots, y_T)$ an explicit response. Distinct from standard diffusion language models or looped transformers that repeatedly applies a shared backbone for all tokens, we propose *contextual loop*, which excutes latent reasoning with large loop depth via only recursively refining its evolving workspace z conditioning on the prompt. After K updates, the completed workspace $z^{\mathrm{S}} = z^{(K)}$ conditions autoregressive generation:

$$\begin{aligned} z^{(k+1)} &= \mathcal{C}_\theta(z^{(k)}, q, \eta_k), \qquad k = 0, \dots, K-1, \\ p_\theta(y \mid q, z^{\mathrm{S}}) &= \prod_{j=1}^{T} p_\theta(y_j \mid q, z^{\mathrm{S}}, y_{<j}). \end{aligned} \tag{1}$$

The control η_k specifies the computation allocated to update k. The shared backbone therefore learns both to construct a latent reasoning context through loops and to use it for explicit decisions.

Continuous dynamics and finite-time computation. There are multiple variants of models with loop structures, including diffusion language model or looped transformer, among which continuous diffusion provides a time-indexed denoising paradigm that is at least as expressive but easier to train compared with a looped transformer (Zhou et al., 2026). With noise at $t = 1$ and a completed workspace at $t = 0$, a conditional velocity field defines

$$\frac{\mathrm{d}z_t}{\mathrm{d}t} = v_\theta(z_t, t \mid q), \qquad z_1 = \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{2}$$

The Euler update $z_{t-h} = z_t - h v_\theta(z_t, t \mid q)$ is a residual contextual loop. Probability-flow ODEs connect continuous diffusion to this transport view (Song et al., 2021), and flow matching learns the conditional velocity directly (Lipman et al., 2023). For a fixed problem and parameter snapshot, the exact flow satisfies $\Phi_{t\to s} = \Phi_{u\to s} \circ \Phi_{t\to u}$ for $t > u > s$. A finite-time map represents the accumulated effect of these dynamics. We parameterize it through an interval-average velocity:

$$F_\theta(z, t, s \mid q) = z - (t - s) u_\theta(z, t, t - s \mid q). \tag{3}$$

The duration $h = t - s$ conditions one evaluation on the interval it should traverse, following flow-map and shortcut formulations (Boffi et al., 2025; Frans et al., 2025; Geng et al., 2025). A student with boundaries $1 = t_0 > \cdots > t_K = 0$ composes K such maps. Loop depth determines the number of evaluations, while map duration determines how much reference computation each evaluation learns to represent. Appendix B develops the corresponding flow identities.

A shared Transformer realization. As an instance, we leverage and largely extend the LaDiR-style paradigm (Kang et al., 2026b), which uses a backbone H_θ, flow projections $P_{\mathrm{f}}, P_{\mathrm{o}}$, and a context

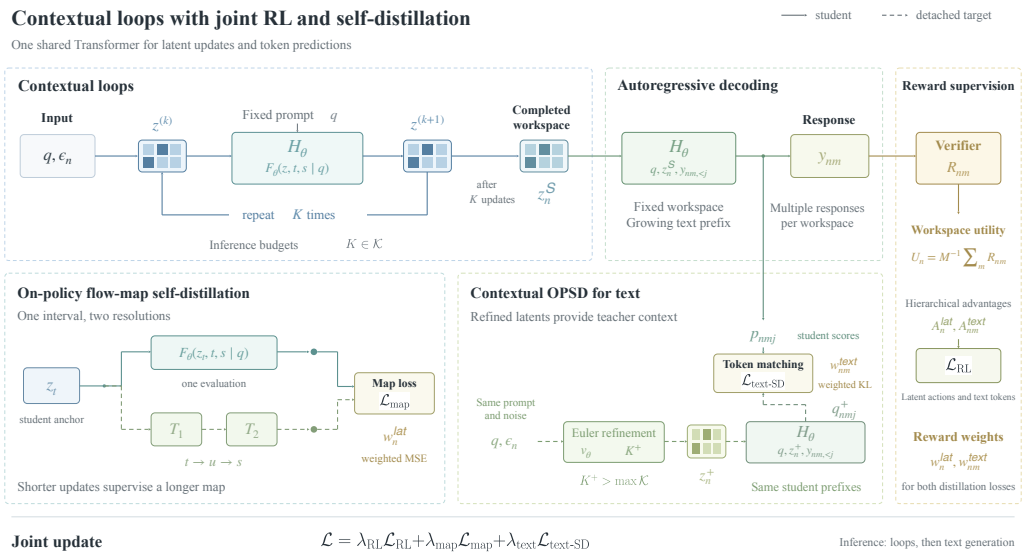


Figure 1: Joint reward-weighted OPSD over latent states and text prefixes. The student constructs a latent workspace and samples explicit responses. Compositions of finer updates supervise its latent flow maps, while a paired workspace constructed with more refinement steps supplies teacher token distributions. Verifier rewards from the shared student rollouts weight both distillation losses, which are jointly optimized with hierarchical RL. Dashed arrows denote stop-gradient targets.

projection P_{r}. The latent branch computes

$$\begin{aligned} H_{\mathrm{lat}} &= H_\theta([E(q), e_{\mathrm{b}}, e(t,h), P_{\mathrm{f}}(z)]; A_{\mathrm{f}})_{\mathrm{latent}}\,, \\ u_\theta(z,t,h \mid q) &= P_{\mathrm{o}}(H_{\mathrm{lat}}, t). \end{aligned} \tag{4}$$

Here $E(q)$ embeds the problem, e_{b} marks the latent context, and $e(t,h) = e_t(t) + e_h(h)$ combines time and duration embeddings. The attention pattern A_{f} allows each latent position to read the problem and the entire workspace. The explicit policy uses causal attention over the problem, the projected endpoint $P_{\mathrm{r}}(z^{\mathrm{S}})$, and the response prefix. The projections connect a pretrained autoregressive backbone to a persistent continuous state, giving it a shared computation that can be applied repeatedly before text generation. Appendix C.1 specifies the architecture.

3.2 Reward-weighted self-distillation over latent and autoregressive states

Latent reasoning with contextual loops not only provides *a more expressive continuous workspace* beyond explicit CoT, but also offers inference-time flexibility by controlling the loop depth: *test-time scaling* in loop steps maximizes the effective reasoning depth, while distillation methods enables high-quality *few-step generation* with minimal latency. We therefore propose a unified post-train framework consists of both reinforcement learning (RL) and on-policy self-distillation (OPSD) to elicite the power of contextual loops.

In particular, both RL and OPSD applies to both the latent branch and the text branch, with shared on-policy student rollouts and reward weights for better efficiency (Section 3.2.1). The two distillation branches supervise complementary predictions along a common reasoning process: the latent branch fits a composed transport endpoint (Section 3.2.2), and the autoregressive branch fits token probabilities supported by the teacher with a better refined workspace (Section 3.2.3). We also provide comprehensive theoretical analysis in Appendix B.

3.2.1 Shared on-policy states and reward weights

A common rollout population. For each problem, a behavior snapshot generates N latent trajectories and samples M responses from each endpoint z_n^{S}. The verifier returns $R_{nm} = r(q, y_{nm})$, and the workspace utility is $U_n = M^{-1}\sum_m R_{nm}$. Intermediate latent states $(z_{n,t}, t)$ provide anchors

for transport supervision, while response prefixes $y_{nm,<j}$ provide states for token supervision. This hierarchical sampling is inspired by LaDi-RL (Kang et al., 2026a), and we further emphasize that its sampled responses supply the rewards for both distillation and RL objective in Section 3.3.

The shared rollout yields the advantages

$$A_n^{\text{lat}} = \text{standardize}_n(U_n), \qquad A_{nm}^{\text{text}} = \text{standardize}_m(R_{nm}). \tag{5}$$

The latent advantage compares workspaces through average downstream reward, and the text advantage compares responses generated from a common workspace. Analogously to Kang et al. (2026a), their clipped policy losses form $\mathcal{L}_{\text{RL}} = \alpha_K \mathcal{L}_{\text{lat-RL}} + (1 - \alpha_K)\mathcal{L}_{\text{text-RL}}$. The anchored stochastic construction of Li et al. (2026) provides the intermediate latent actions used to score finite maps. Appendix C.3 specifies the policy losses and budget-dependent weighting.

Reward weighting across the latent–text hierarchy. For temperature $\tau_r > 0$, define

$$a_n = N \frac{\exp(U_n/\tau_r)}{\sum_i \exp(U_i/\tau_r)}, \qquad b_{nm} = M \frac{\exp(R_{nm}/\tau_r)}{\sum_{\ell=1}^{M} \exp(R_{n\ell}/\tau_r)}. \tag{6}$$

The mean-one factor a_n prioritizes workspaces by their downstream utility, and b_{nm} further distinguishes responses produced from a common workspace, thus verifier feedback guides both the latent computation and the explicit predictions. The latent weight w_n^{lat} derives from a_n, and the text weight w_{nm}^{text} derives from $a_n b_{nm}$. Each family is floored and normalized within the prompt; distillation is active for prompt groups containing a positively rewarded workspace. Algorithm 2 gives the complete weighting rule. Exponential weighting connects this allocation to entropy-regularized reward selection (Peters et al., 2010; Peng et al., 2019), as developed in Appendix C.2.

3.2.2 On-policy flow-map self-distillation

Composed targets at student states. We sample a target duration h independently of the student budget K and select a student anchor (z_t, t) with $t \geq h$. Splitting $h = h_1 + h_2$ gives an intermediate time $u = t - h_1$ and endpoint time $s = t - h$. A parent map traverses $t \to s$ in one evaluation, supervised by the composition

$$\widehat{z}_s = \text{sg}[T_2(T_1(z_t, t, u \mid q), u, s \mid q)]\,. \tag{7}$$

sg denotes stop-gradient. At the finest target interval, the child operators are Euler steps under an exponential moving average (EMA) of the local-velocity model. Coarser targets compose shorter online flow maps, which follows a finer reference computation from the sampled student state.

Reward-weighted endpoint matching. The latent self-distillation objective is

$$\mathcal{L}_{\text{map}} = \mathbb{E}\left[w_n^{\text{lat}} \frac{\|F_\theta(z_t, t, s \mid q) - \widehat{z}_s\|_F^2}{Ld} \right], \tag{8}$$

where the expectation includes prompts, student trajectories, target durations, and eligible anchors. The parent prediction learns to reproduce the endpoint of several finer updates, with reward weights emphasizing workspaces that support successful responses.

Independent sampling of the target duration exposes student states to supervision at multiple temporal resolutions. A learned parent map can subsequently serve as a child in a longer composition, allowing progressively larger intervals to absorb the effect of finer computation. The resulting hierarchy connects multi-step reference dynamics to few-step generation; see Appendix B.1 for more analysis.

3.2.3 Contextual OPSD for autoregressive reasoning

Constructing privileged latent context. Let $\mathcal{K}$ be the set of student budgets. For the same problem and initial noise as trajectory n, the online model constructs a paired workspace z_n^+ with $K^+ > \max \mathcal{K}$ local-velocity number of function evaluations (NFEs), and the corresponding teacher scores q_{nmj}^+:

$$\begin{aligned} z_n^+ &= \text{sg}[\text{Euler}_{K^+}(v_\theta; \epsilon_n, q)]\,, \\ q_{nmj}^+ &= \text{sg}\big[p_\theta(\cdot \mid q, z_n^+, y_{nm,<j})\big]\,. \end{aligned} \tag{9}$$

Algorithm 1 Joint reinforcement and self-distillation over latent and autoregressive states

Require: LaDiR-initialized parameters θ; local-velocity EMA $\bar{\theta}$; verifier r; student budgets $\mathcal{K}$; teacher budget K^+.

1: **for** each training iteration **do**
2: Sample problems and a student budget $K \in \mathcal{K}$; independently sample target duration h.
3: Generate N latent trajectories and M responses per workspace; evaluate R_{nm}.
4: Compute workspace utilities, latent–text advantages, and distillation weights.
5: Construct composed latent targets and paired-context token targets using Algorithm 2.
6: Evaluate $\mathcal{L}_{\mathrm{RL}}$, $\mathcal{L}_{\mathrm{map}}$, and $\mathcal{L}_{\text{text-SD}}$ on the shared rollout.
7: Update θ with Equation (11); update the local-velocity EMA $\bar{\theta}$.
8: **end for**
9: **return** A shared latent–text model with inference budgets $K \in \mathcal{K}$.

Recall the student scores $p_{nmj} = p_\theta(\cdot \mid q, z_n^{\mathrm{S}}, y_{nm,<j})$. Here the teacher and the student share model parameters, initial noises, and sampled texts. The diffence lies in the computations of the latent workspaces: the refined workspace supplies privileged context for the teacher, extending contextual OPSD (Zhao et al., 2026) through additional latent reasoning.

Reward-weighted token matching. Let $\mathcal{I}$ index valid sampled token positions in active distillation groups. The autoregressive OPSD objective is

$$\mathcal{L}_{\text{text-SD}} = \frac{\sum_{(q,n,m,j)\in\mathcal{I}} w_{nm}^{\text{text}} D_{\mathrm{KL}}\big(\mathcal{B}_j(q_{nmj}^+) \| \mathcal{B}_j(p_{nmj})\big)}{\sum_{(q,n,m,j)\in\mathcal{I}} w_{nm}^{\text{text}}}. \tag{10}$$

The operator $\mathcal{B}_j$ groups vocabulary probabilities using a common teacher-defined partition, specified in Appendix B.2.1. The teacher distribution and both latent endpoints are held fixed, and the student prediction updates the shared decoder and context projection. The loss transfers token predictions supported by the refined latents to the student's context at its own prefixes.

Latent computation as a supervision dimension. Remarkably, the reasoning workspace serves both as answer context and as a source of self-distillation targets. Refinement depth controls the computation used to construct the teacher context, and workspace size controls its representational capacity. This makes latent compute a design dimension of OPSD: additional reasoning shapes the teacher distribution, and contextual distillation trains the explicit policy to use a workspace produced at a smaller student budget. Appendix B.2.2 relates this transfer to teacher quality and the accuracy of token predictions along student prefixes.

3.3 Joint optimization and flexible inference

Starting from an initialization of a latent reasoning model (such as LaDiR (Kang et al., 2026b)), we jointly optimize the latents and text responses with both RL and OPSD losses:

$$\mathcal{L} = \lambda_{\mathrm{RL}}\mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{map}}\mathcal{L}_{\mathrm{map}} + \lambda_{\text{text}}\mathcal{L}_{\text{text-SD}}. \tag{11}$$

Reinforcement assigns task credit to latent actions and generated tokens, flow-map self-distillation compresses the workspace transitions, and contextual OPSD trains the explicit predictions conditioned on their endpoints. The objectives share rollouts, verifier rewards, and backbone parameters, coupling latents generation to the policy that interprets it as training changes the state distribution.

The two distillation losses express a common reward-weighted fitting principle in continuous and categorical prediction spaces. Endpoint regression fits composed latent updates; token matching fits the contextual teacher's distribution. Reward weights allocate fitting capacity across the corresponding student states. Appendix C.2 develops this conditional-projection perspective, and Algorithm 1 summarizes the joint update.

We now summarize our training procedure, which provides a practical route to reasoning models with flexible loop depths. Representation supervision first supplies an initial interface for continuous reasoning, while joint reinforcement and distillation adapt its task utility and the computation used to

construct it. Finer latent updates and refined contexts continually supply targets for shorter computations, and the budget curriculum (details in Appendix D.3) extends this supervision toward fewer loop evaluations, training a pretrained autoregressive backbone to perform contextual-loop reasoning across several inference budgets. In particular, flow-map compression reduces the latent computation required for each response, and contextual OPSD transfers the explicit policy's predictions from a more extensively refined workspace. These mechanisms connect the cost of generating a context to the quality of the response it supports. Appendix D.5 analyzes their contribution to compute per verified solution.

At inference, the model composes K deterministic (ODE) or stochastic (SDE) maps and generates the response from their endpoint. A backbone with D_{tr} layers executes KD_{tr} layers of latent computation over these loops, with the workspace carrying state between evaluations. Training across map durations makes this effective depth selectable through $K \in \mathcal{K}$.

4 Experiments

4.1 Experimental Setup

Model and initialization. We use Qwen3-8B-Base (Yang et al., 2025) for latent reasoning and autoregressive answer generation. All methods within a domain start from the same supervised checkpoint, which also serves as that domain's SFT reference. Coding experiments use a checkpoint based on $125{,}000$ steps of SFT on filtered Ling-Coder data. Mathematical reasoning uses this checkpoint after additional supervised training on $14{,}791$ mathematical examples retained after decontamination and length filtering. This initialization provides the latent representation and answer policy used by subsequent post-training. Architecture, representation learning, and implementation details appear in Appendix C; the initialization and optimization settings are collected in Table 5.

Training data and comparisons. Coding post-training uses $24{,}000$ Python problems from AceCode-87K and KodCode-V1, with reward equal to the fraction of unit tests passed. Mathematical post-training uses $30{,}656$ problems derived from DeepScaleR and binary verification of the extracted final answer. We include diffusion language models LLaDA (Nie et al., 2025) and DREAM (Ye et al., 2025) as baselines. For contextual loop models, we compare four configurations: SFT initialization, GRPO, flow map, and MeanFlow. Here GRPO denotes our implementation of LaDi-RL-style hierarchical *joint latent–text GRPO* (Kang et al., 2026a): both policies are updated from verifier rewards. We found for omitting text OPSD loss yields slightly better results for flow-map (results w/o text OPSD are marked with †), but Meanflow always performs better with text OPSD. By default, flow map adds reward-weighted latent map distillation with $\lambda_{\mathrm{text}} = 0$; MeanFlow uses the MeanFlow-style latent objective and contextual text OPSD with $\lambda_{\mathrm{text}} = 0.5$. Appendix D further specifies the rollout groups, objectives, and hyperparameters.

Benchmarks and metrics. The reported coding benchmarks are HumanEval, HumanEval+, MBPP+, and MBPP, using EvalPlus; MBPP uses the MBPP+ problem subset with its base tests. The reported mathematical benchmarks are MATH-500 and Minerva Math. We present pass@1 and multi-candidate pass@k, which measure individual-candidate accuracy and candidate-set coverage, respectively. Our main comparison uses an RL budget of 1000 updates; Table 1 reports each method's best recorded checkpoint for each benchmark within this budget. Workspace-content and latent-budget ablations compare interventions at fixed checkpoints.

Evaluation settings. Our evaluation protocol draws 16 candidates per problem with a 30-step Euler ODE latent sampler and shared answer-decoding settings. This fixes the latent inference budget across the compared checkpoints. Answers have a 768-token limit, and method comparisons use matched problem sets. The SFT reference is the shared initialization. On MBPP+, we test the contribution of the generated workspace by replacing it before decoding, and separately vary the latent budget over 30, 8, 4, 2, and 1 steps. Both ablations hold the checkpoint and answer-decoding settings fixed. Appendix D specifies the adopted protocol, hyperparameters, and interventions; Appendix E presents the additional results.

Table 1: Coding pass@1 (%): best recorded checkpoint within the training budget for each method and benchmark. Bold marks the highest displayed score in each column.

Method	HumanEval	HumanEval+	MBPP+	MBPP
LLaDA	35.4	30.5	42.1	50.1
Dream	56.7	50.0	57.4	68.7
LaDiR (SFT)	66.0	62.6	42.3	48.5
LaDi-RL (Joint GRPO)	70.9	67.0	58.0	67.9
LaDi-OPSD (Flow map†)	72.4	68.2	57.4	67.2
LaDi-OPSD (MeanFlow)	**73.7**	**68.8**	**60.6**	**70.9**

Table 2: Coding pass@5 and pass@10 (%) at the training-budget endpoint. Bold marks the highest displayed score in each column.

	HumanEval		HumanEval+		MBPP+		MBPP	
Method	pass@5	pass@10	pass@5	pass@10	pass@5	pass@10	pass@5	pass@10
LaDiR (SFT)	86.1	89.4	**82.0**	**85.8**	60.4	65.3	68.2	73.5
LaDi-RL (Joint GRPO)	82.6	85.9	78.2	81.5	70.2	72.9	78.7	81.4
LaDi-OPSD (Flow map†)	84.6	88.2	81.0	84.4	68.6	72.0	78.3	81.1
LaDi-OPSD (MeanFlow)	**87.4**	**90.9**	81.7	85.1	**70.3**	**73.0**	**81.9**	**84.4**

4.2 Main Results

We compare SFT initialization, joint GRPO, flow-map self-distillation, and MeanFlow under the training budget and evaluation settings in Section 4.1. The coding results assess both single-candidate accuracy and success with multiple candidates. The workspace ablation then examines the contribution of the latent context to answer generation.

Coding accuracy. Table 1 reports the best recorded pass@1 within the training budget for each method and benchmark. Flow map reaches 72.4% on HumanEval and 68.2% on HumanEval+, exceeding joint GRPO by 1.5 and 1.2 percentage points, respectively. MeanFlow reaches 68.5% on MBPP, compared with GRPO's 68.1%. All three post-training configurations improve over the shared SFT initialization on the four coding benchmarks.

Success with multiple candidates. Table 2 reports pass@5 and pass@10. MeanFlow raises HumanEval pass@5 from GRPO's 82.6% to 87.4% and pass@10 from 85.9% to 90.9%. On MBPP, it raises the corresponding scores from 78.7% to 81.9% and from 81.4% to 84.4%. All four gains have paired 95% bootstrap intervals above zero. Flow map also improves over GRPO on HumanEval and HumanEval+, reaching 84.6%/88.2% and 81.0%/84.4%, respectively. These results show improved candidate-set coverage under the common sampling and decoding budget, suggesting that generation diversity is preserved in our distillation.

Accuracy with few-step latent budgets. Table 3 varies the latent budget while fixing answer decoding. Flow-map MBPP+ pass@1 ranges from 56.9% to 57.2% over the five budgets – even at one step, flow map attains 57.2%. The stable accuracy strongly support the effectiveness of our distillation methods in few-step generation in the latent space, which finishes our initial target of *powerful and efficient* latent reasoning model with as few as only one contextual loop.

Mathematical reasoning. Table 4 reports the mathematical pass@k comparison to multiple candidates on MATH-500 and Minerva Math. On MATH-500, flow map without text OPSD improves over GRPO by 5.9, 8.8, and 10.4 percentage points at pass@5, pass@10, and pass@16. On Minerva Math, the corresponding improvements are 2.7, 3.2, and 3.3 points.

Table 3: Latent-budget ablation on MBPP+: pass@1 (%) at fixed checkpoints. The 30-step column uses the common Euler route; smaller budgets use learned interval maps for flow map. Answer decoding is fixed.

Configuration	30 steps	8 steps	4 steps	2 steps	1 step
LaDi-OPSD (Flow map†)	57.0	57.0	56.9	56.9	57.2

Table 4: Candidate coverage (%) on MATH-500 and Minerva Math. All rows use the adopted 30-step ODE protocol; flow map uses text-OPSD coefficient 0. An upward arrow marks a positive paired 95% interval relative to GRPO.

	MATH-500			Minerva Math		
Configuration	pass@5	pass@10	pass@16	pass@5	pass@10	pass@16
LaDiR (SFT)	27.1	37.2	44.4	4.8	7.5	9.6
LaDi-RL (Joint GRPO)	22.4	29.3	34.2	5.3	7.7	9.6
LaDi-OPSD (Flow map†)	**28.3**	**38.1**	**44.6**	**8.0**	**10.9**	**12.9**

5 Discussion

A training route to recurrent latent computation. Representation supervision gives an autoregressive backbone an interface for continuous reasoning states, supplied here by LaDiR. Joint RL and OPSD then adapt its generated workspace to verifier rewards and train its updates across interval lengths. These signals offer a route from a pretrained LLM to a recurrent latent reasoner. A backbone with D_{tr} layers executes approximately KD_{tr} layers of latent computation over K loops. Theory relates such feedback to iterative algorithmic expressivity under the architectural assumptions of looped Transformers (Saunshi et al., 2025).

Latent reasoning expands the design of privileged context. A latent workspace conditions both the answer policy and its contextual teacher. Improving the representation through reward optimization can therefore benefit inference and self-teaching together. Teacher refinement depth determines how its context is computed; workspace width determines how many latent positions represent the problem; student loop depth controls deployment cost. These choices give OPSD additional compute and representation dimensions, whose value depends on teacher advantage and transfer accuracy.

Multi-step refinement and few-step generation. Shorter latent updates supervise longer maps, and the refined teacher workspace supervises student token predictions. These targets connect multi-step refinement to few-step generation: the same model can spend more evaluations on its workspace or use a smaller trained budget to construct its answer context. The results in Table 3 suggest that our method enables *loop reasoning models without recursive loop* – like a shallow expanse of water reflecting a boundless sky, a finite sequence of learned updates can carry the imprint of a much longer computation.

6 Conclusion

We conceptually connect various latent reasoning models through the perspective of contextual loop. To enable expressive yet efficient latent reasoning, we propose joint reward-weighted OPSD that trains an LLM on both the latent states it refines and the text prefixes it generates. Flow-map targets compress workspace construction, and refined latent contexts supervise the explicit policy that reads the resulting workspace. This connects the two objectives through a shared reasoning process and makes latent computation a controllable source of privileged context. LaDi-OPSD realizes the approach through joint RL and OPSD from LaDiR initialization, providing a route to latent reasoning with selectable loop budgets. Our method outperforms other latent reasoning models on both coding and mathematical reasoning datasets with controlled or less computation.

AI USE STATEMENT

Generative AI assisted with literature discovery and writing (manuscript drafting and polishing) only. and LaTeX preparation. We have not used generative AI tools for other tasks. We take responsibility for the final content of this work.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning, Generative Modeling and LLM Reasoning. There are some potential societal consequences of our work, none which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

Algorithms 1–3 specify training, target construction, and inference. Appendix B gives the assumptions and proofs for the theoretical analysis; Appendices C and D specify implementation details (including the architecture and the algorithm) and experimental settings (including optimization, sampling, evaluation recipe and hyper-parameters), respectively.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to Ponder. *arXiv preprint arXiv:2107.05407*, 2021. URL https://arxiv.org/abs/2107.05407.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training Diffusion Models with Reinforcement Learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14f75513f0f1ca01de1e826b52e6b840-Paper-Conference.pdf.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 33346–33382. Curran Associates, Inc., 2025. doi: 10.52202/085713-1121. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2ff3605c9e442d55156d99fe4f5f228d-Paper-Conference.pdf.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dOPSD: On-Policy Self-Distillation for Diffusion Language Models. *arXiv preprint arXiv:2607.04428*, 2026. URL https://arxiv.org/abs/2607.04428.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit Chain of Thought Reasoning via Knowledge Distillation. *arXiv preprint arXiv:2311.01460*, 2023. URL https://arxiv.org/abs/2311.01460.

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. URL https://arxiv.org/abs/2405.14838.

Liangliang Dong, Lianlei Shan, and Shuaimin Li. Self-SoftCoT: A self-consistent framework via position-aware latent space reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32393–32414. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1496. URL https://aclanthology.org/2026.acl-long.1496/.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped Transformers for Length Generalization. In *International Conference on Learning Representations*,

2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/25cc3adf8c85f7c70989cb8a97a691a7-Paper-Conference.pdf.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the Gap Between Latent and Explicit Reasoning with Looped Transformers. *arXiv preprint arXiv:2606.31779*, 2026. URL https://arxiv.org/abs/2606.31779.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One Step Diffusion via Shortcut Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/559a0998fab1d19b80e7e43a5852401c-Paper-Conference.pdf.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 41340–41391. Curran Associates, Inc., 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Paper-Conference.pdf.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean Flows for One-step Generative Modeling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 75460–75482. Curran Associates, Inc., 2025. doi: 10.52202/085713-2534. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6d13e085b79d454da5910e4ca82a3d9d-Paper-Conference.pdf.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as Programmable Computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.08933.

Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016. URL https://arxiv.org/abs/1603.08983.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ac015d409635f196f9e3e9dcfb9a94e-Paper-Conference.pdf.

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. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2412.06769.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation. *arXiv preprint arXiv:2601.20802*, 2026. URL https://arxiv.org/abs/2601.20802.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Yi-An Ma, and Lianhui Qin. LaDi-RL: Latent Diffusion Reasoning Prevents Entropy Collapse in Reinforcement Learning. *arXiv preprint arXiv:2602.01705*, 2026a. URL https://arxiv.org/abs/2602.01705.

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*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/843e1d8ab4819c8c8a4221fce2c8d455-Paper-Conference.pdf.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c204d12afa0175285e5aac65188808b4-Paper-Conference.pdf.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM Improves Controllable Text Generation. In *Advances in Neural Information Process-*

ing Systems, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022. doi: 10.52202/068431-0313. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1be5bc25d50895ee656b8c2d9eb89d6a-Paper-Conference.pdf.

Zhiqi Li, Wen Zhang, and Bo Zhu. Flow-Map GRPO: Reinforcement Learning for Few-Step Flow-Map Generators via Anchored Stochastic Composition. *arXiv preprint arXiv:2607.00535*, 2026. URL https://arxiv.org/abs/2607.00535.

Haitao Lin, Peiyan Hu, Minsi Ren, Zhifeng Gao, Zhi-Ming Ma, Guolin Ke, Tailin Wu, and Stan Z Li. On the Design of One-step Diffusion via Shortcutting Flow Paths. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b78c69286221bbce7d12f98b5b45f337-Paper-Conference.pdf.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training Flow Matching Models via Online RL. In *Advances in Neural Information Processing Systems*, volume 38, pp. 40783–40818. Curran Associates, Inc., 2025. doi: 10.52202/085713-1362. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3a10c46572628d58cb44fb705f25cbbf-Paper-Conference.pdf.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference. *arXiv preprint arXiv:2310.04378*, 2023. URL https://arxiv.org/abs/2310.04378.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50608–50646. Curran Associates, Inc., 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/48b383b24230e0e6e649d9c98dae4d8c-Paper-Conference.pdf.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *arXiv preprint arXiv:1910.00177*, 2019. URL https://arxiv.org/abs/1910.00177.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative Entropy Policy Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010. doi: 10.1609/aaai.v24i1.7727. URL https://ojs.aaai.org/index.php/AAAI/article/view/7727.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. CANDI: Hybrid Discrete-Continuous Diffusion Models. *arXiv preprint arXiv:2510.22510*, 2025. URL https://arxiv.org/abs/2510.22510.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align Your Flow: Scaling Continuous-Time Flow Map Distillation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 146459–146512. Curran Associates, Inc., 2025. doi: 10.52202/085713-4900. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d79ac139911df25d27f14bbf008deaee-Paper-Conference.pdf.

Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00512.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. CRISP: Compressed Reasoning via Iterative Self-Policy Distillation. *arXiv preprint arXiv:2603.05433*, 2026. URL https://arxiv.org/abs/2603.05433.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with Latent Thoughts: On the Power of Looped Transformers. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2676109d49d1eb26d6bc584a8f556305-Paper-Conference.pdf.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-Distillation Enables Continual Learning. *arXiv preprint arXiv:2601.19897*, 2026. URL https://arxiv.org/abs/2601.19897.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2011.13456.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think Silently, Think Fast: Dynamic Latent Compression of LLM Reasoning Chains. In *Advances in Neural Information Processing Systems*, volume 38, pp. 4646–4668. Curran Associates, Inc., 2025. doi: 10.52202/085713-0164. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0706261aedab63814a2b73c32564b4c4-Paper-Conference.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT++: Test-Time Scaling with Soft Chain-of-Thought Reasoning. *arXiv preprint arXiv:2505.11484*, 2025b. URL https://arxiv.org/abs/2505.11484.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin

Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yin_One-step_Diffusion_with_Distribution_Matching_Distillation_CVPR_2024_paper.html.

Guibin Zhang, Jiayang Lyu, Ran Sun, Xinlei Yu, Haoyu Zhao, Qibing Ren, and Shuicheng Yan. Latent On-Policy Self-Distillation. *arXiv preprint arXiv:2608.13040*, 2026. URL https://arxiv.org/abs/2608.13040.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. *arXiv preprint arXiv:2601.18734*, 2026. URL https://arxiv.org/abs/2601.18734.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously Augmented Discrete Diffusion model for Categorical Generative Modeling. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/39b77b5e422b4e070e2811b73ea9bcf7-Paper-Conference.pdf.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary Continuous Discrete Diffusion: Make Your Diffusion Language Model a Latent Reasoner. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=80nuPd5Gvb.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling Latent Reasoning via Looped Language Models. *arXiv preprint arXiv:2510.25741*, 2025. URL https://arxiv.org/abs/2510.25741.

A Extended Related Work

Learning continuous reasoning representations. Chain-of-thought prompting expresses intermediate reasoning as text (Wei et al., 2022). Several approaches transfer this supervision into continuous representations. Implicit CoT distills a teacher's explicit reasoning into hidden states across network layers (Deng et al., 2023), while gradual internalization removes intermediate textual steps during training (Deng et al., 2024). CODI jointly trains explicit and implicit reasoning, aligning their hidden states at a designated token (Shen et al., 2025). SoftCoT learns a projection from assistant-generated soft thoughts into a frozen language model's input space (Xu et al., 2025a); SoftCoT++ diversifies these thoughts through specialized initial tokens and contrastive learning to support parallel exploration (Xu et al., 2025b). These methods establish complementary ways to learn a continuous interface for reasoning: layerwise transfer, progressive internalization, hidden-state alignment, and projected soft thoughts. Such representation supervision provides an initialization on which task rewards and self-distillation can subsequently operate.

Expressivity and allocation of recurrent depth. Adaptive Computation Time and PonderNet learn how many recurrent updates to allocate to an input (Graves, 2016; Banino et al., 2021). Constructive analyses show that looped Transformers can implement iterative programs (Giannou et al., 2023) and simulate classes of chain-of-thought computations under specified architectural assumptions (Saunshi et al., 2025). Learned loops also improve length generalization on tasks with iterative solutions (Fan et al., 2025). At the level of inference strategies, Snell et al. (2025) study how problem difficulty determines the effective allocation of test-time compute. These perspectives distinguish the capacity supplied by repeated computation from the decision of how much computation to spend. A contextual flow map introduces a further control: the duration represented by each update. Loop count and map duration jointly determine how a fixed inference budget traverses the latent dynamics.

Continuous, discrete, and hybrid diffusion for language. Diffusion-LM generates text through continuous word embeddings (Li et al., 2022), DiffuSeq develops conditional sequence generation with diffusion (Gong et al., 2023), and LLaDA scales masked-token denoising to language modeling (Nie et al., 2025). Recent hybrid approaches combine continuous geometry with discrete token structure. CADD augments discrete denoising with a paired continuous latent process (Zheng et al., 2026); CANDI separates discrete identity corruption from continuous geometric corruption (Pynadath et al., 2025). CCDD jointly denoises continuous representations and discrete tokens, and analyzes the expressivity and trainability of continuous diffusion relative to discrete diffusion and looped Transformers (Zhou et al., 2026). These hybrids couple latent and token states throughout denoising. Contextual loops provide a sequential interface between the two spaces: latent refinement constructs a workspace, and autoregressive decoding then uses its completed state. This interface exposes both the workspace trajectory and the conditioned response prefixes as states for learning.

Transport foundations and distillation objectives. Probability-flow ODEs associate continuous diffusion with deterministic transport (Song et al., 2021). Flow matching and rectified flow learn vector fields that transport a reference distribution to a target distribution (Lipman et al., 2023; Liu et al., 2023). Within accelerated generation, Latent Consistency Models learn few-step generators in image latent spaces (Luo et al., 2023), and Consistency Trajectory Models learn maps between intermediate points on probability-flow trajectories (Kim et al., 2024). Distribution Matching Distillation fits the generated distribution through a distribution-level objective (Yin et al., 2024). Recent work on shortcutting flow paths separates the mathematical constraints on valid shortcuts from component choices in their training (Lin et al., 2026). Together, these approaches expose several possible distillation targets, including endpoint predictions, intermediate transport maps, and generated distributions. Our composed endpoint targets preserve a direct correspondence between a coarse latent update and the finer computation it represents.

Student-state supervision and the source of teacher context. DAgger studies supervision under the state distribution induced by the learner (Ross et al., 2011); MiniLLM uses reverse-KL distillation for generative language models (Gu et al., 2024). Contextual self-distillation additionally varies the information available to the teacher. SDFT conditions the teacher on demonstrations to support continual learning (Shenfeld et al., 2026), and SDPO converts environmental feedback into token-level targets through a feedback-conditioned self-teacher (Hübotter et al., 2026). CRISP uses a

conciseness instruction to produce teacher distributions and compress explicit reasoning on student rollouts (Sang et al., 2026). These examples show how the teacher's context determines which behavior is transferred at a visited prefix. In contextual-loop reasoning, the refinement budget controls the computation invested in that context, and the workspace size determines the capacity of its representation. Pairing student and teacher workspaces through a common problem and initial noise provides a correspondence between the contexts used for supervision across computation budgets.

Reward optimization across latent and textual policies. GRPO uses group-relative rewards to optimize reasoning policies (Shao et al., 2024). DDPO casts diffusion denoising as sequential decision-making (Black et al., 2024); Flow-GRPO develops online reward optimization for flow-matching models (Liu et al., 2025), and Flow-Map GRPO introduces anchored stochastic composition for few-step generators (Li et al., 2026). Within latent reasoning, CoLaR combines controllable compression with reinforcement over latent trajectories (Tan et al., 2025), while Self-SoftCoT learns position-aware latent transformations through group sequence policy optimization (Dong et al., 2026). Relative Entropy Policy Search and advantage-weighted regression connect reward-sensitive policy fitting to constrained improvement and weighted regression (Peters et al., 2010; Peng et al., 2019). This connection motivates allocating distillation weight according to downstream utility. In the latent–text hierarchy, rewards averaged over responses estimate workspace utility, and individual response rewards distinguish textual realizations. Applying these weights to both endpoint regression and token matching directs self-distillation toward latent computations and explicit predictions associated with successful solutions.

B Theoretical Analysis

We derive two bounds tied to the objectives optimized by LaDi-OPSD. Proposition 1 relates reward-weighted endpoint fitting to the workspace produced by a reference computation and its readout utility. Proposition 2 relates the implemented partitioned token loss to transfer of a contextual teacher's reward. Their method-specific content is the connection between the sampled supervision measures, the fitted quantities, and task reward; the proofs use standard stability and probability inequalities.

Setting. Fix the behavior law and the parameter snapshots used for evaluation and detached targets. A vectorized workspace lies in $\mathbb{R}^D$, $D = Ld$, with Euclidean norm equal to its Frobenius norm. Expectations below are population quantities for the stated problem, initialization, and sampling laws. The empirical objectives estimate these risks; the bounds apply when their population errors and coverage conditions hold.

Reference transport. Let $\Phi_{t\to s}$ be the unique flow of a continuous velocity field $v(z, t \mid q)$ that is locally Lipschitz in state, with solutions existing on the intervals considered. Time decreases from noise at 1 to a completed workspace at 0. For fixed q, the standard flow identities (Boffi et al., 2025) are

$$\Phi_{t\to t}(z) = z, \qquad \Phi_{t\to s} = \Phi_{u\to s} \circ \Phi_{t\to u}, \qquad s \le u \le t, \tag{12}$$

with interval-average velocity

$$\Phi_{t\to t-h}(z) = z - hu^{\Phi}(z, t, h), \qquad u^{\Phi}(z, t, h) = \frac{1}{h}\int_{t-h}^{t} v(\Phi_{t\to r}(z), r)\,\mathrm{d}r. \tag{13}$$

These follow by restarting and integrating the same ODE solution. They specify the reference computation compressed by the learned maps.

B.1 From endpoint fitting to workspace and readout error

Fix a grid $1 = t_0 > \cdots > t_K = 0$ whose evaluated intervals have positive probability under the discrete training interval sampler. Let F_k, T_k, and Φ_k be the evaluated student map, its fixed composed teacher, and a fixed reference map on interval k. The reference may be the composed teacher itself or a map implementing a finer reference computation. Their input x includes the problem, workspace, and target-construction controls. Let μ_k be the behavior anchor law conditional on that interval, and let $W_k \ge 0$ include relative reward weights and prompt-level confidence or

selection, with $0 < \mathbb{E}W_k < \infty$. Expectations defining μ_k and W_k are conditional on the interval. Define

$$w_k(x) = \frac{\mathbb{E}[W_k \mid X_k = x]}{\mathbb{E}W_k}, \qquad \widetilde{\mu}_k(\mathrm{d}x) = w_k(x)\mu_k(\mathrm{d}x). \tag{14}$$

The targets are deterministic given this enlarged input and their fixed parameter snapshots. Let ν_k be the input law along the evaluated student trajectory, with the same conditional target-control rule.

Proposition 1 (Endpoint fitting and workspace error)**.** *Assume $\nu_k \ll \widetilde{\mu}_k$ and $\mathrm{d}\nu_k/\mathrm{d}\widetilde{\mu}_k \le \chi_k < \infty$. Suppose Φ_k is L_k-Lipschitz on the domains containing the coupled trajectories, uniformly over problems and target-construction controls. Define the finite errors*

$$\varepsilon_k = \mathbb{E}_{\widetilde{\mu}_k}[\|F_k - T_k\|^2 / D], \qquad \eta_k = \left(\mathbb{E}_{\nu_k} \|T_k - \Phi_k\|^2\right)^{1/2}. \tag{15}$$

For student and reference trajectories with the same problem, initial workspace, and target-construction controls,

$$\left(\mathbb{E}\left\|z^{\mathrm{S}} - z^{\mathrm{ref}}\right\|^2\right)^{1/2} \le B_{\mathrm{lat}} := \sum_{k=0}^{K-1} \left(\sqrt{D\chi_k\varepsilon_k} + \eta_k\right) \prod_{\ell=k+1}^{K-1} L_\ell. \tag{16}$$

For a fixed readout whose bounded expected verifier reward $U(q, z)$ is uniformly L_U-Lipschitz in z, $|\mathbb{E}U(q, z^{\mathrm{S}}) - \mathbb{E}U(q, z^{\mathrm{ref}})| \le L_U B_{\mathrm{lat}}$.

Proof. The density bound implies $\mathbb{E}_{\nu_k} \|F_k - T_k\|^2 \le D\chi_k\varepsilon_k$. In $F_k(z_k^{\mathrm{S}}) - \Phi_k(z_k^{\mathrm{ref}})$, insert $T_k(z_k^{\mathrm{S}})$ and $\Phi_k(z_k^{\mathrm{S}})$. Minkowski's inequality gives

$$e_{k+1} \le L_k e_k + \sqrt{D\chi_k\varepsilon_k} + \eta_k, \qquad e_k = \left(\mathbb{E}\left\|z_k^{\mathrm{S}} - z_k^{\mathrm{ref}}\right\|^2\right)^{1/2}. \tag{17}$$

Unroll from $e_0 = 0$. The utility statement follows from the Lipschitz assumption and Cauchy–Schwarz. □

The interval-conditioned contribution to Equation (8) is $(\mathbb{E}W_k)\varepsilon_k$. Thus its reward-weight normalizer is part of the conversion to ε_k. If $\mathrm{d}\nu_k/\mathrm{d}\mu_k \le C_k$ and $w_k \ge \kappa_k > 0$ on the relevant support, then $\chi_k \le C_k/\kappa_k$. This sufficient condition separates visitation coverage from reward reweighting. Evaluating the same anchor population as the behavior law gives $C_k = 1$; deterministic inference, stochastic training, replay, prompt confidence, and response selection enter through their actual laws.

The insight here is that the loss controls agreement with its actual teacher. Taking $\Phi_k = T_k$ makes $\eta_k = 0$: subject to the stated coverage and teacher-stability conditions, the population endpoint loss directly controls the difference between a student rollout and a rollout of its composed targets. This is the quantity trained by map self-distillation. Choosing a separate, more accurate reference retains η_k as the teacher's own approximation error. The distinction identifies the role of the finest-scale Euler targets without attributing an accuracy guarantee to the composition loss alone.

A sufficient readout-stability condition. Let $U(q, z) = \mathbb{E}_{y\sim p(\cdot|q,z)} r(q, y)$ for a fixed decoder, reward $r \in [0, 1]$, and maximum response horizon H. If every live prefix a satisfies $\mathrm{TV}(p(\cdot \mid q, z, a), p(\cdot \mid q, z', a)) \le L_p \|z - z'\|$, a stepwise maximal coupling yields

$$|U(q, z) - U(q, z')| \le \min\{1, HL_p \|z - z'\|\}. \tag{18}$$

Thus one may take $L_U = HL_p$. The condition applies to the actual decoder used in U, including any sampling transformation. The contextual-transfer result below instead controls token disagreement directly on student prefixes.

Relation to the budget sweep. Table 3 shows small accuracy changes across latent budgets for distilled models. Proposition 1 identifies sufficient conditions for a compressed map to preserve a reference computation. The reported sweep measures budget sensitivity; composition residuals, teacher errors, and trajectory-coverage factors provide the complementary diagnostics for quantifying this mechanism. In particular, the bounds concern fidelity to a chosen reference, whose task utility is assessed separately.

B.2 Theory of Contextual Reward Transfer

The second component of our analysis links a refined latent teacher to the explicit policy trained on student continuations. Our main result, Proposition 2, uses the implemented top-k objective, the unresolved tail mass, and coverage of student prefixes. It gives an explicit condition for preserving a contextual teacher's task advantage at a smaller latent-computation budget.

B.2.1 The vocabulary-partitioned objective

At each prefix let Q denote the teacher distribution, p the student's full-vocabulary distillation distribution, and S the teacher's top-k set, with a fixed tie-breaking rule. Let $O = \mathcal{V} \setminus S$. The common partition retains each token in S and aggregates O:

$$\mathcal{B}(Q) = ((Q_v)_{v\in S}, Q_O), \qquad \mathcal{B}(p) = ((p_v)_{v\in S}, p_O), \quad Q_O = \sum_{v\in O} Q_v, \quad p_O = \sum_{v\in O} p_v. \tag{19}$$

The training divergence is

$$D_{\mathrm{KL}}(\mathcal{B}(Q)\|\mathcal{B}(p)) = \sum_{v\in S} Q_v \log \frac{Q_v}{p_v} + Q_O \log \frac{Q_O}{p_O}. \tag{20}$$

The standard KL chain rule gives

$$D_{\mathrm{KL}}(Q\|p) = D_{\mathrm{KL}}(\mathcal{B}(Q)\|\mathcal{B}(p)) + Q_O D_{\mathrm{KL}}(Q(\cdot \mid O)\|p(\cdot \mid O)). \tag{21}$$

For $Q_O = 0$, the final term is defined as zero; for $Q_O > 0, p_O = 0$, both full and partitioned KL are infinite. Otherwise the identity follows by writing $Q_v = Q_O Q(v \mid O)$ and $p_v = p_O p(v \mid O)$ within the tail. Refining a nested partition increases its KL toward the full-vocabulary value.

A tail-mass inequality. A useful consequence for our objective is

$$\mathrm{TV}(Q, p) \le \mathrm{TV}(\mathcal{B}(Q), \mathcal{B}(p)) + \min\{Q_O, p_O\}. \tag{22}$$

Indeed, the difference between the full and partitioned total variations is $\frac{1}{2}(\sum_{v\in O} |Q_v - p_v| - |Q_O - p_O|)$, which is at most $\min\{Q_O, p_O\}$. This elementary inequality allows a reward bound using the optimized divergence and aggregate tail probabilities, even when conditional tail KL is large. The tail term is sharp: for $0 < \gamma < 1/2$, distributions $(1-\gamma, \gamma, 0)$ and $(1-\gamma, 0, \gamma)$ have identical top-one partitions and total variation γ.

B.2.2 Reward transfer on weighted student prefixes

The evaluation and supervision populations. Fix a population of paired contexts $(q, z^{\mathrm{S}}, z^{+})$. Let Q be the teacher's full-vocabulary autoregressive policy conditioned on z^{+}, p the student's distillation policy conditioned on z^{S}, and π its deployed sampling policy. The distributions Q, p use the distillation temperature; π includes the evaluation temperature and any nucleus filtering. Each policy uses the same maximum horizon H and absorbing termination convention. For reward $r(q, y) \in [0, 1]$, let $V(P)$ be the expected reward of fresh continuations from policy P on this fixed context population.

For T valid token positions, including a sampled termination token, define $\ell_\pi = \mathbb{E}_\pi T \in (0, H]$ and

$$d_\pi(A) = \frac{1}{\ell_\pi} \mathbb{E}_\pi \left[\sum_{j=1}^{T} \mathbf{1}\{X_j \in A\} \right], \qquad X_j = (q, z^{\mathrm{S}}, z^{+}, j, y_{<j}). \tag{23}$$

For training, let $\mathcal{J}$ be a nonempty replay batch of valid token occurrences before response filtering. Let $W_i \ge 0$ include the relative response weight, prompt confidence, and acceptance mask for token

i. To match the implemented reduction in Equation (34), define

$$d_{\mathrm{b}}(A) = \mathbb{E}_{\mathrm{b}}\left[\frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}} \mathbf{1}\{X_i \in A\}\right],$$
$$\zeta = \mathbb{E}_{\mathrm{b}}\left[\frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}} W_i\right], \tag{24}$$
$$d_w(A) = \frac{1}{\zeta}\mathbb{E}_{\mathrm{b}}\left[\frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}} W_i\mathbf{1}\{X_i \in A\}\right].$$

The expectation includes batch sampling, and we assume $0 < \zeta < \infty$. This defines probability measures while retaining the random valid-token denominator, outcome-dependent weights, and filtering. If $d_i = D_{\mathrm{KL}}(\mathcal{B}(Q_i)\|\mathcal{B}(p_i))$, the exact population relation is

$$\mathcal{L}^{\mathrm{impl}}_{\mathrm{text,pop}} := \mathbb{E}_{\mathrm{b}}\left[\frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}} W_i d_i\right] = \zeta\,\mathbb{E}_{d_w} d_i. \tag{25}$$

The contextual-transfer bound. We now present the second central result.

Proposition 2 (Reward transfer from the implemented token objective)**.** *Assume $d_\pi \ll d_w$ and $\mathrm{d}d_\pi/\mathrm{d}d_w \le \Gamma < \infty$. At each prefix use the same teacher-defined partition for Q and p, and define*

$$\delta_{\mathrm{AR}} = \mathbb{E}_{d_w} D_{\mathrm{KL}}(\mathcal{B}(Q)\|\mathcal{B}(p)) < \infty,$$
$$\tau_O = \mathbb{E}_{d_w}\min\{Q_O, p_O\}, \qquad \nu_{\mathrm{samp}} = \mathbb{E}_{d_\pi}\,\mathrm{TV}(p,\pi). \tag{26}$$

Then

$$|V(Q) - V(\pi)| \le B_{\mathrm{AR}} := \min\left\{1,\ \ell_\pi\left[\nu_{\mathrm{samp}} + \sqrt{\frac{\Gamma\delta_{\mathrm{AR}}}{2}} + \Gamma\tau_O\right]\right\}. \tag{27}$$

For a reference policy evaluated on the same context population, write $\Delta = V(Q) - V(\pi_{\mathrm{ref}})$. Then

$$V(\pi) - V(\pi_{\mathrm{ref}}) \ge \Delta - B_{\mathrm{AR}}. \tag{28}$$

Proof. Consider the $H+1$ hybrid policies that use π for their first j positions and Q thereafter, $j = 0, \ldots, H$. For $j = 1, \ldots, H$, compare hybrids $j-1$ and j: they share the distribution of their first $j-1$ positions under π. Given such a live prefix, the expected reward after the next token, followed by Q, is a function of that token in $[0,1]$. Thus the difference between the hybrids' rewards is bounded by the expected total variation of their next-token distributions. Telescoping gives

$$|V(Q) - V(\pi)| \le \mathbb{E}_\pi \sum_{j=1}^{T} \mathrm{TV}(Q_j, \pi_j) = \ell_\pi \mathbb{E}_{d_\pi}\,\mathrm{TV}(Q,\pi). \tag{29}$$

By the triangle inequality, Equation (22), and Pinsker's inequality,

$$\mathrm{TV}(Q,\pi) \le \mathrm{TV}(p,\pi) + \sqrt{\tfrac{1}{2}D_{\mathrm{KL}}(\mathcal{B}(Q)\|\mathcal{B}(p))} + \min\{Q_O, p_O\}. \tag{30}$$

Take expectation under d_π, use Jensen's inequality on the square root, and use the density bound on each nonnegative d_w-controlled term. This gives the bracket in Equation (27). Bounded reward caps the difference at one. Subtracting $V(\pi_{\mathrm{ref}})$ from $V(\pi) \ge V(Q) - B_{\mathrm{AR}}$ gives Equation (28). □

The agreement term is $\delta_{\mathrm{AR}} = \mathcal{L}^{\mathrm{impl}}_{\mathrm{text,pop}}/\zeta$; τ_O uses only the teacher and student aggregate tail probabilities already required by that loss. Their contributions quantify transfer from the refined context on student prefixes. The factor ℓ_π reflects the expected number of explicit decisions, while Γ accounts for reward selection and changes in the paired-context and prefix populations. If $\mathrm{d}d_\pi/\mathrm{d}d_{\mathrm{b}} \le C$ and $\mathrm{d}d_w/\mathrm{d}d_{\mathrm{b}} \ge \kappa > 0$, then $\Gamma \le C/\kappa$. The value $C = 1$ requires identical token-occupancy laws, including context selection, sampling, and length normalization. Retaining ζ separates improved token agreement from a smaller loss caused by lower confidence or acceptance mass. Relative weight floors control supervision within groups; prompt confidence and response selection enter the complete measure d_w. The coverage condition specifies transfer to the evaluation population.

When π samples directly from p, $\nu_{\mathrm{samp}} = 0$. When π only renormalizes p on a nucleus set, their per-prefix total variation is exactly the discarded probability mass. Different decoding temperatures contribute to the general ν_{samp} term. A full-vocabulary partition gives $\tau_O = 0$.

C Implementation Details

This appendix specifies the architecture and training algorithms.

C.1 Architecture and latent representation

Shared backbone and projections. The backbone in Section 4.1 serves both latent transport and autoregressive readout. Separate flow and readout multilayer perceptrons project latent coordinates to the Transformer hidden dimension. A transport head applies time-conditioned adaptive layer normalization and projects latent hidden states back to the workspace dimension. A sinusoidal representation followed by an MLP encodes time; an additive duration embedding conditions the finite-map branch as in Equation (4). The duration embedding's output layer starts at zero, so the initialized finite-map predictor inherits the instantaneous-velocity computation. Table 5 specifies the workspace size and time discretization.

Attention and decoding. Prompt tokens use causal attention. During transport, each latent position attends to the prompt, conditioning tokens, and the entire workspace. During text generation, the completed workspace is inserted between learned boundary embeddings, and the Transformer uses causal attention over that prefix and generated responses. The vocabulary head operates on text predictions; the transport head operates on latent hidden states. The local-velocity branch uses the time embedding, and the finite-map branch additionally uses the duration embedding.

Supervised representation learning. Following LaDiR (Kang et al., 2026b), a frozen VAE supplies encoded reasoning blocks during supervised initialization. Let z_0 be such a block and $\epsilon \sim \mathcal{N}(0, I)$. The initialization uses

$$z_t = (1-t)z_0 + t\epsilon, \qquad v^\star = \epsilon - z_0, \qquad \mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\|v_\theta(z_t, t \mid q) - v^\star\|_F^2 / (Ld)\right]. \tag{31}$$

A supervised text objective trains workspace-conditioned generation. Projectors are first aligned to a frozen backbone and then jointly adapted with it. During RL, the transport model directly generates the workspace. The domain-specific checkpoints and their training data are recorded in Appendix D.2.

C.2 Joint objectives, reward weights, and teachers

Shared rollouts and gradient paths. The joint update follows Algorithm 1 and Equation (11). The backbone, projectors, and transport head are optimized through the applicable reinforcement and distillation branches. Anchors, sampled actions, rewards, advantages, and targets are detached during replay. Text OPSD holds both latent contexts fixed and updates the student decoder and context projection. The detailed target-construction and inference procedures are given in Algorithms 2 and 3.

Relative weights and prompt confidence. The mean-one factors a_n, b_{nm} are defined in Equation (6). The reported launch configurations implement a soft-confidence variant of the positive-group selection described in Section 3.2.1. With $u_q = \mathrm{clip}(\max_n U_n, 0, 1)$, define

$$g_q^{\mathrm{lat}} = f_{\mathrm{lat}} + (1 - f_{\mathrm{lat}})u_q, \qquad g_q^{\mathrm{text}} = f_{\mathrm{text}} + (1 - f_{\mathrm{text}})u_q. \tag{32}$$

The two branches use separate absolute confidence floors $f_{\mathrm{lat}}, f_{\mathrm{text}}$. Within each prompt, relative weights are first floored and normalized; confidence is applied afterwards:

$$\begin{aligned} w_n^{\mathrm{lat}} &= g_q^{\mathrm{lat}} \frac{\max\{a_n, w_0^{\mathrm{lat}}\}}{N^{-1}\sum_i \max\{a_i, w_0^{\mathrm{lat}}\}}, \\ w_{nm}^{\mathrm{text}} &= g_q^{\mathrm{text}} \frac{\max\{a_n b_{nm}, w_0^{\mathrm{text}}\}}{(NM)^{-1}\sum_{i,j} \max\{a_i b_{ij}, w_0^{\mathrm{text}}\}}. \end{aligned} \tag{33}$$

Table 6 distinguishes absolute confidence floors from relative weight floors. The binary-selection variant is recovered by setting both confidences to $\mathbf{1}\{\max_n U_n > 0\}$. Every valid token of a response receives its response weight; an empty weighted token set contributes zero loss. The population weights in Appendix B include the chosen confidence rule and any response filter.

Text loss reduction and response filtering. Let $\mathcal{J}$ contain all valid token occurrences in a replay batch before the optional response filter. Let $A_{nm} \in \{0, 1\}$ denote acceptance, with $A_{nm} = 1$ in unfiltered variants, and let d_{nmj} be the partitioned KL in Equation (10). The soft-confidence implementation preserves the absolute confidence scale by using

$$\mathcal{L}_{\text{text}}^{\text{impl}} = \frac{1}{|\mathcal{J}|} \sum_{(q,n,m,j)\in\mathcal{J}} w_{nm}^{\text{text}} A_{nm} d_{nmj}. \tag{34}$$

The denominator includes valid tokens from every response before acceptance filtering. If $\widehat{\zeta} = |\mathcal{J}|^{-1} \sum_{\mathcal{J}} w_{nm}^{\text{text}} A_{nm} > 0$, this equals $\widehat{\zeta}$ times the weight-normalized KL on accepted tokens. This distinguishes the normalized fitting objective in Equation (10) from its confidence-scaled implementation. Zero accepted weight gives zero loss. Appendix B.2.2 retains this normalization when relating the implemented loss to reward transfer.

Interpretation of reward weighting. Exponential selection has the standard entropy-regularized interpretation (Peters et al., 2010; Peng et al., 2019):

$$\frac{a_n}{N} = \omega_n^\star, \qquad \omega^\star = \arg\max_{\omega\in\Delta_N} \left\{ \sum_n \omega_n U_n - \tau_r D_{\text{KL}}(\omega \| \operatorname{Unif}_N) \right\}. \tag{35}$$

The objective equals $\tau_r \log(N^{-1} \sum_n e^{U_n/\tau_r}) - \tau_r D_{\text{KL}}(\omega \| \omega^\star)$. Thus the relative weights allocate supervision toward higher observed utility. Prompt confidence separately controls the total weight assigned to a prompt.

Conditional fitting. For student input X, detached weight $W \geq 0$, endpoint target Y, and full-vocabulary teacher Q^+, the standard conditional-projection identity characterizes the fitting objectives. Assume $0 < \mathbb{E}W < \infty$ and $\mathbb{E}[W \|Y\|^2] < \infty$, and work on inputs with $a(X) = \mathbb{E}[W \mid X] > 0$.

Lemma 1 (Weighted conditional fitting)**.** *The unrestricted minimizers of* $\mathbb{E}[W \|f(X) - Y\|^2 / D]$ *and* $\mathbb{E}[W D_{\text{KL}}(Q^+ \| p(\cdot \mid X))]$ *are*

$$f_W^\star(X) = \frac{\mathbb{E}[WY \mid X]}{a(X)}, \qquad p_W^\star(\cdot \mid X) = \frac{\mathbb{E}[WQ^+(\cdot) \mid X]}{a(X)}. \tag{36}$$

Proof. Conditioning on X, expand the square about $f_W^\star$; its cross term is zero. In the categorical risk, the teacher-entropy term is constant and the remaining cross-entropy is minimized by $p_W^\star$. These give the two identities under the usual extended-value KL conventions. □

For a deterministic target $Y = T(X)$, weighting allocates finite model capacity while the unrestricted optimum remains $T(X)$. The categorical identity applies to full-vocabulary matching. Our implemented top-k partition depends on the privileged-context teacher and can vary at fixed student input; its task-reward analysis uses the pointwise partition identities in Appendix B.2.1.

Teacher roles. An EMA of trainable parameters supplies the local-velocity teacher at the finest target duration, updated after each optimizer step by $\bar{\theta} \leftarrow \beta\bar{\theta} + (1 - \beta)\theta$. Coarser targets compose shorter maps with detached online parameters. The contextual teacher uses online parameters for both its higher-budget Euler workspace and its token predictions. Thus the EMA supplies the finest-scale local-velocity targets, composed children supervise interval compression, and the paired context supplies explicit-token targets.

Endpoint and velocity implementations. For child durations h_1, h_2 and detached predictions u_1, u_2 evaluated successively, $\widehat{z}_s = z_t - h_1 u_1 - h_2 u_2$. Hence

$$F_\theta(z_t, t, s) - \widehat{z}_s = -h \left(u_\theta(z_t, t, h) - \frac{h_1 u_1 + h_2 u_2}{h} \right), \qquad h = h_1 + h_2. \tag{37}$$

Endpoint regression therefore multiplies the squared average-velocity residual by h^2; dividing the endpoint loss by h^2 implements the duration-normalized alternative. Displacements and residual differences use full-precision arithmetic.

C.3 LATENT EXPLORATION AND POLICY REPLAY

Local anchored stochastic composition. For a finite map from t to $0 < s < t \leq 1$, first compute its deterministic endpoint

$$\widetilde{z}_s = F_\theta(z_t, t, s \mid q). \tag{38}$$

Following the local anchored construction of Li et al. (2026), take $\delta_0 > 0$, define $\delta = \min(\delta_0, s/2)$ and $a = s - \delta$, and choose an exploration strength $\lambda_{\mathrm{a}} > 0$. Under our noise-to-data time convention, the Gaussian parameters are

$$\mu_\theta = \widetilde{z}_s - \delta\lambda_{\mathrm{a}}^2 \left(\frac{\widetilde{z}_s}{1-a} + v_\theta(\widetilde{z}_s, s \mid q) \right), \tag{39}$$

$$\sigma^2 = 2\delta\lambda_{\mathrm{a}}^2 \frac{a}{1-a}, \qquad z_s \sim \mathcal{N}(\mu_\theta, \sigma^2 I). \tag{40}$$

This is the local Gaussian approximation to the anchored conditional, with $0 < a < s < 1$ and strictly positive variance. The approximation is applied at selected intermediate transitions. The kernel's model dependence enters through the endpoint and its local velocity, and policy replay differentiates through both terms. The variance is determined by the sampling schedule and exploration settings. The terminal transition to $s = 0$ uses the deterministic map.

We select a contiguous stochastic window within eligible intermediate transitions, with the window settings in Table 5. For a one-loop trajectory, the initial Gaussian noise supplies sample diversity, and the single map reaches the terminal workspace directly. That iteration updates the text policy, finite-map objective, and contextual text objective. Inference composes the deterministic maps from sampled initial noise.

Hierarchical advantages and replay ratios. We use the hierarchical latent–text reinforcement algorithm of LaDi-RL (Kang et al., 2026a). The behavior factorization is

$$z_{n,1} \sim \mathcal{N}(0, I), \qquad z_{n,t_{k+1}} \sim \pi_{\theta_{\mathrm{b}}}^{\mathrm{lat}}(\cdot \mid z_{n,t_k}, q), \qquad y_{nm} \sim \pi_{\theta_{\mathrm{b}}}^{\mathrm{text}}(\cdot \mid q, z_n^{\mathrm{S}}). \tag{41}$$

Selected intermediate transitions use the stochastic kernel in Equation (40); the remaining transitions use the deterministic map. For a replay ratio ρ and detached advantage A, let $a \in \{\mathrm{lat}, \mathrm{text}\}$ index the policy. Its clipped minimization objective is

$$\ell_{\mathrm{clip}}^a(\rho, A) = -\min\{\rho A, \mathrm{clip}(\rho, 1-\epsilon_-^a, 1+\epsilon_+^a)A\}. \tag{42}$$

The latent policy uses equal widths $\epsilon_-^{\mathrm{lat}} = \epsilon_+^{\mathrm{lat}}$, and the text policy uses its configured lower and upper widths. Averaging over scored latent actions and valid text positions yields

$$\begin{aligned} \mathcal{L}_{\mathrm{RL}} = \alpha_K \mathbb{E}_{\mathrm{lat}}[\ell_{\mathrm{clip}}^{\mathrm{lat}}(\rho^{\mathrm{lat}}, A^{\mathrm{lat}})] \\ + (1-\alpha_K)\mathbb{E}_{\mathrm{text}}[\ell_{\mathrm{clip}}^{\mathrm{text}}(\rho^{\mathrm{text}}, A^{\mathrm{text}})]. \end{aligned} \tag{43}$$

For a group $(x_1, \ldots, x_G)$, define

$$\mathrm{standardize}(x_i) = \frac{x_i - \overline{x}}{\max\left\{\sqrt{G^{-1}\sum_j (x_j - \overline{x})^2}, \epsilon_A\right\}}. \tag{44}$$

This gives the advantages in Equation (5). Rewards, advantages, sampled transitions, and sampled response tokens are detached during replay.

For latent dimension $D = Ld$, let the score of an observed action under Equation (40) be

$$\ell_\theta^{\mathrm{lat}} = \frac{1}{D}\sum_{a=1}^{D} \left[-\frac{(z_{s,a} - \mu_{\theta,a})^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2) \right]. \tag{45}$$

The ratio used by the instantiated surrogate is

$$\rho^{\mathrm{lat}} = \exp(\ell_\theta^{\mathrm{lat}} - \ell_{\theta_{\mathrm{b}}}^{\mathrm{lat}}) = \left(\frac{\pi_\theta^{\mathrm{lat}}(z_s \mid z_t, q)}{\pi_{\theta_{\mathrm{b}}}^{\mathrm{lat}}(z_s \mid z_t, q)} \right)^{1/D}. \tag{46}$$

This is a dimension-tempered surrogate: the genuine joint-density importance ratio is $(\rho^{\mathrm{lat}})^D$. The normalization sets the scale of clipping for a high-dimensional workspace and is part of the chosen optimization objective. Deterministic transitions contribute through the distillation objectives and shared parameters; Gaussian policy scores apply to the selected stochastic actions.

The text behavior policy $\pi_\theta^{\mathrm{text}}$ applies the configured temperature and nucleus filter to the decoder probabilities. Replay uses the same transformation, giving

$$\rho_{nmj}^{\mathrm{text}} = \frac{\pi_\theta^{\mathrm{text}}(y_{nmj} \mid q, z_n^{\mathrm{S}}, y_{nm,<j})}{\pi_{\theta_{\mathrm{b}}}^{\mathrm{text}}(y_{nmj} \mid q, z_n^{\mathrm{S}}, y_{nm,<j})}. \tag{47}$$

The denominator is positive for sampled behavior tokens. A token excluded by the current nucleus has numerator zero. Importance-sampling interpretations additionally require support coverage of the policy being evaluated by the behavior policy; the replay expression above specifies the clipped surrogate on stored actions. The text self-distillation distributions use the temperature-scaled, full-vocabulary probabilities in Equation (9). Latent and text policy losses average Equation (42) over their respective scored actions, using the clipping widths in Table 5. The coefficient α_K takes its configured value when latent actions are scored and is zero for one-loop iterations; the latent term is defined as zero when its scored-action set is empty. Valid response masks include the sampled termination token and retain all generated tokens of a response that reaches its length cap.

Replay and numerical precision. Responses are grouped by length and replayed in chunks. One student backbone pass supplies hidden states for text reinforcement and contextual distillation, and vocabulary projection is processed in token windows. Model states use BF16; Gaussian scores, probability reductions, map residuals, and EMA master updates use FP32. Prompt representations can be cached across latent updates.

Algorithm 2 Reward-weighted latent and autoregressive OPSD targets

Require: Student rollouts $\{\epsilon_n, (z_{n,t})_t, z_n^{\mathrm{S}}, y_{nm}, R_{nm}\}$ for problem q; target duration h; online parameters θ; EMA $\bar{\theta}$; teacher budget K^+.

Compute $U_n = M^{-1}\sum_m R_{nm}$ and branch confidences $g_q^{\mathrm{lat}}, g_q^{\mathrm{text}}$ from Equation (32).
Set $a_n = N\,\mathrm{softmax}_n(U_n/\tau_r)$, $b_{nm} = M\,\mathrm{softmax}_m(R_{nm}/\tau_r)$.
Obtain w_n^{lat} and w_{nm}^{text} by flooring a_n and $a_n b_{nm}$, respectively, normalizing each family to mean one, and multiplying by its branch confidence, as in Equation (33).
for each trajectory n **do**
  if the map objective is enabled and $w_n^{\mathrm{lat}} > 0$ **then**
    Sample an eligible anchor (z_t, t); partition $h = h_1 + h_2$; set $u = t - h_1$, $s = t - h$.
    if h is the finest supported target duration **then**
      $z_u \leftarrow z_t - h_1 v_{\bar{\theta}}(z_t, t \mid q)$.
      $\widehat{z}_s \leftarrow z_u - h_2 v_{\bar{\theta}}(z_u, u \mid q)$.
    else
      $z_u \leftarrow F_\theta(z_t, t, u \mid q)$; $\widehat{z}_s \leftarrow F_\theta(z_u, u, s \mid q)$.
    end if
  end if
  if text OPSD is enabled and $g_q^{\mathrm{text}} > 0$ **then**
    $z_n^+ \leftarrow \mathrm{Euler}_{K^+}(v_\theta; \epsilon_n, q)$.
    for each response m and its valid prediction positions j **do**
      $q_{nmj}^+ \leftarrow p_\theta(\cdot \mid q, z_n^+, y_{nm,<j})$.
      Store the teacher's top-k partition and probabilities, including its tail mass.
    end for
    Score teacher–student log-likelihood margins; form response acceptance A_{nm} if the verified filter is enabled.
  end if
end for
return Detached map and token targets, reward weights, and acceptance masks for Equations (8) and (34).

C.4 Algorithms for paired targets and inference

Algorithm 2 expands the OPSD step in Algorithm 1. All targets and reward weights are computed under stop-gradient. The latent branch predicts a composed endpoint, and the AR branch predicts a refined-context token distribution on the sampled continuation. Algorithm 3 exposes the loop budget as an inference-time control.

Algorithm 3 Budgeted contextual-loop reasoning

Require: Problem q; trained model θ; loop budget K; time boundaries $1 = t_0 > \cdots > t_K = 0$.
1: Sample initial workspace $z \sim \mathcal{N}(0, I)$.
2: **for** $k = 0, \ldots, K-1$ **do**
3: $\quad z \leftarrow F_\theta(z, t_k, t_{k+1} \mid q)$.
4: **end for**
5: Decode an explicit response autoregressively from $p_\theta(\cdot \mid q, z)$ under the selected sampling rule.
6: **return** Explicit solution y.

D Experimental Details

We provide more experimental details including the data, hyperparameters, and evaluation protocol in this section.

D.1 Corpora, filtering, and verifiers

Coding data. The coding RL corpus combines $13{,}118$ problems from AceCode-87K and $10{,}882$ from KodCode-V1. Construction selects executable tests, validates reference solutions, and applies quality and duplicate filtering. Reward is the fraction of independently scored test units passed, with the execution timeout in Table 5.

Mathematical data. The RL corpus derives from DeepScaleR. The original $30{,}671$-record training split has a separate 500-problem holdout. The final training version removes six reviewed malformed records and nine benchmark-overlap cases, giving the $30{,}656$ examples used in Section 4.1; 27 prompts additionally receive the standard answer-format suffix. Exact and 13-word overlap candidates are reviewed against the pinned evaluation sets. Mathematical reward uses binary equivalence checking of the extracted final answer, with the extraction and verification rules in Appendix D.4.

The initialization corpus starts from $24{,}000$ worked solutions from R1-Distill-SFT. Conservative overlap and prompt filtering yields $23{,}804$ records; token-length filtering leaves the $14{,}791$ examples used for math SFT. The SFT overlap screen covers MATH-500, AIME 2024, AIME 2025, AMC 2023, Minerva Math, OlympiadBench, and the RL holdout. The filtering stages distinguish the source-corpus size from the final training artifact.

D.2 Initialization and optimization settings

Coding runs share an initialization based on 125k steps of SFT on filtered Ling-Coder data. The math initialization continues this checkpoint for 500 SFT updates on the filtered mathematical corpus. This produces one shared starting checkpoint for all RL methods within each domain. During RL, ground-truth flow and text SFT losses have zero coefficient. Table 5 collects the common model, rollout, and optimizer settings; Table 6 specifies distillation and method-dependent settings.

Compared training configurations. The SFT initialization provides the common reference. GRPO is our LaDi-RL-style joint latent–text implementation, with both policies optimized by the hierarchical verifier-reward objective and all distillation coefficients set to zero. The reported GRPO numbers come from these local runs. Flow map without text OPSD adds the endpoint objective using independently sampled target intervals, EMA local-velocity targets at the finest scale, and composed child-map targets at coarser scales; its text coefficient is zero. MeanFlow uses the boundary/interior mixture and finite-difference derivatives in Table 6, together with text OPSD coefficient 0.5. Its text targets use the soft-confidence weights without response verification filtering. These configurations

Table 5: Shared model, rollout, and optimization settings for the reported comparisons.

Group	Setting	Value
Model	Backbone / answer decoder	Qwen3-8B-Base
	Frozen SFT VAE encoder / decoder	Qwen3-4B-Base / Qwen3-8B-Base
	Latent workspace / time indices	64×2560 / 1000
Math SFT	Updates / batch size	500 / 32
	Learning rate / warm-up updates	5×10^{-6} / 25
Rollouts	Latents per prompt / answers per latent	16 / 5
	Prompts / answers per optimizer update	8 / 640
	Latent / text RL coefficient	0.9 / 0.1; full text RL for one-loop rollouts
	Latent / text clip parameters	0.01 / $(0.2, 0.28)$
	Advantage stabilizer ϵ_A	10^{-4}
	Latent score reduction	Mean over workspace coordinates
	KL coefficient / policy epochs	0 / 1
	Stochastic window	Up to 2 transitions within $[0, 5)$; deterministic final transition
	Rollout answer decoding	Temperature 1, top-$p = 0.98$, cap 768
	Coding verifier timeout	10 seconds
Optimization	Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$
	Learning rate / schedule	10^{-5}; 100 linear warm-up updates, then constant
	Weight decay / gradient norm clip	0.01 / 1
	Minibatches per optimizer update	1
	Hardware / precision	8×H100; BF16 with FP32 loss arithmetic
	Distribution	Parameter sharding; one prompt per worker
	RL comparison budget	1000 optimizer updates
	Independent training seeds	One per reported trajectory

share the domain-specific SFT checkpoint and RL corpus. The coding comparisons include all four configurations; the mathematical comparisons report SFT, GRPO, and flow map.

D.3 Computation budgets and curricula

Independent behavior and target schedules. Let $\mathcal{K} = (8, 4, 2, 1)$ and let U_j be uniform over its first j entries. With optimizer index i starting at zero, phase length S, and phase $j \in \{1, 2, 3\}$, set $x = (i - (j-1)S)/S$ and $c(x) = (1 - \cos(\pi x))/2$. During that phase the behavior distribution is

$$P_i = (1 - c(x))U_j + c(x)U_{j+1}. \tag{48}$$

The behavior phase length is given in Table 6; the schedule becomes uniform over all four budgets at update 900.

The target schedule uses its separate phase length, interpolating $U_1 \to U_2 \to U_3$ and then $U_3 \to Q_\star$. Its terminal mixture balances expected exposure across the configured horizon. Accounting for the discrete sum $\sum_{i=0}^{S-1} c(i/S) = (S-1)/2$ gives $Q_\star \approx (0.2241, 0.2425, 0.2607, 0.2728)$ for the target settings in Table 6. At the balancing horizon, target sampling becomes uniform. Behavior and target draws use independent seeds and are shared across distributed workers within an optimizer step. A behavior budget K sets trajectory length; a target budget B sets the regression duration $h = 1/B$. An anchor is drawn uniformly from stored states with $t \geq h$.

Time discretization. Student finite-map inference uses exact integer boundaries on the time grid. A target interval is split into positive integer durations summing to the parent. For example, the 125-tick interval at $B = 8$ uses 62- and 63-tick children; Equation (37) weights their velocity predictions by these actual durations. The contextual teacher applies the online local velocity on its finer Euler grid. This construction keeps rollout depth, target interval, and contextual teacher depth as separate controls.

Table 6: Method-specific distillation and exploration settings. Absolute prompt-confidence floors and relative within-prompt weight floors act at different stages of Equation (33).

Setting	Value
RL / each enabled latent distillation coefficient	1 / 1
Text OPSD coefficient	0 or 0.5, as identified in each result
Rollout budgets / curriculum phase	$\{8, 4, 2, 1\}$ / 300 updates
Flow-map target budgets	$\{8, 4, 2, 1\}$
Target curriculum phase / balancing horizon	100 / 3000 updates
Finest-scale target teacher	Local-velocity EMA, decay 0.999
Coarse composition teacher	Detached online child maps
Contextual text teacher	Online weights, stop-gradient, 16 Euler steps
Text distillation temperature / partition	1 / teacher top-64 plus tail bucket
Text loss denominator	All valid replay tokens, before response filtering
Relative reward temperature (both branches)	1
Relative floors $(w_0^{\mathrm{lat}}, w_0^{\mathrm{text}})$	$(0.1, 0.1)$
Prompt-confidence floors $(f_{\mathrm{lat}}, f_{\mathrm{text}})$	$(0.05, 0)$
MeanFlow boundary / interior mixture	0.75 / 0.25
MeanFlow finite-difference step	0.01
GRPO baseline stochastic sampler	CPS: noise 0.8, diversity guidance 0.8
Finite-map stochastic sampler	Local ASFMC: strength 0.7, anchor radius 0.03, diversity guidance 0
CPS / ASFMC transition score	Unscaled negative MSE / Gaussian log density
Unified rollout time grid	Unshifted Euler or integer interval grid
Legacy behavior-route timestep shift	3; separate from the unified rollout grid

Table 7: Evaluation populations under the adopted protocol. Coding tests use EvalPlus 0.3.1; the version column identifies the benchmark data. Plus variants use extended tests on the same problems.

Benchmark	Problems	Data version	Grading
HumanEval / HumanEval+	164 each	0.1.10	Base / extended tests
MBPP / MBPP+	378 each	0.2.0	Base / extended tests
MATH-500	500	Pinned snapshot	Final-answer equivalence
Minerva Math	272	Pinned snapshot	Final-answer equivalence

D.4 Evaluation protocol and ablation design

Benchmarks and grading. Table 7 specifies the reported evaluation populations. Coding evaluation uses EvalPlus 0.3.1. MBPP uses the MBPP+ problem subset with base tests, so both versions evaluate the same problems. HumanEval solutions are evaluated as standalone candidate programs against the base or extended test suite.

For mathematics, the repository's verifier extracts the last boxed answer, prioritizing text after the reasoning delimiter, or an answer following an explicit final-answer prefix. It then applies normalized, numerical, and bounded symbolic equivalence checks. The fallback that accepts an arbitrary trailing number is disabled. These extraction rules apply to both mathematical rewards and evaluation.

Adopted sampling and decoding protocol. All main results use 30 Euler ODE latent steps with classifier-free guidance scale 1. Answer decoding uses temperature 1, top-$p = 0.98$, top-$k = 50$, and a 768-token cap. Each problem receives 16 independent candidates, generated in batches of 8 with seed 1234 and random streams keyed by problem and sample-batch offset. The ablations below retain these answer-decoding settings and change only the stated workspace intervention or latent-step budget.

Workspace-content intervention. For each candidate, we first generate its normal workspace $z \in \mathbb{R}^D$ using the adopted 30-step route. Immediately before answer decoding, the workspace is

kept, replaced by zero, or replaced by

$$\widetilde{z} = \sigma(z)\epsilon, \qquad \sigma(z) = \sqrt{\|z\|^2 / D}, \qquad \epsilon \sim \mathcal{N}(0, I_D). \tag{49}$$

The Gaussian intervention scales independent noise by the original candidate's workspace RMS. Prompt, checkpoint, workspace dimensions, and answer-decoding settings remain fixed. Latent and text random streams are controlled separately, so drawing replacement noise leaves the text seed unchanged. The MBPP+ comparison in Table 8 evaluates SFT initialization and fixed joint-GRPO and flow-map checkpoints. It measures how answer accuracy responds to altering the completed workspace at each fixed checkpoint.

Latent-budget ablation. At the same coding checkpoints, Table 3 varies the latent-step count over 30, 8, 4, 2, and 1. The 30-step reference uses the common Euler route. At smaller budgets, flow map uses learned interval maps, and SFT and GRPO use Euler steps. The checkpoint is held fixed across the sweep, so the table measures its inference behavior at each requested budget.

Metrics and uncertainty. For $n = 16$ independently sampled candidates with c correct, we use the standard estimator

$$\widehat{\text{pass@}}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \qquad k \in \{1, 5, 10, 16\}, \tag{50}$$

where $\binom{n-c}{k} = 0$ when $n - c < k$. We use 2000 paired bootstrap resamples of problems and 95% percentile intervals for score differences. An upward arrow in a fixed-checkpoint method comparison marks an interval above zero relative to its GRPO reference. For the workspace intervention, $\uparrow$ and $\downarrow$ compare each replacement with the same model's normal workspace. Each caption states its comparison reference. Table 1 reports per-benchmark checkpoint maxima and uses bold values to identify the highest displayed scores. Each method contributes one training trajectory, so the intervals quantify problem-sampling uncertainty.

D.5 Computation and theory-linked diagnostics

For a response of length T at latent budget K, separate map computation, decoder prefill, and token generation:

$$C(q; K, T) = KC_{\text{map}} + C_{\text{pre}}(q, L) + \sum_{j=1}^{T} C_{\text{tok}}(q, L, j). \tag{51}$$

At fixed response length and comparable map cost, the approximate total speed ratio for $K_0 \to K$ is $(K_0 C_{\text{map}} + C_{\text{text}})/(KC_{\text{map}} + C_{\text{text}})$, where C_{text} is prefill plus decoding cost. Latency measurement additionally fixes hardware, batching, precision, cache reuse, and warm-up. Equation (52) specifies how per-attempt cost combines with success probability. The present tables report accuracy, coverage, and budget sensitivity; measured end-to-end latency is a separate quantity.

Compute per verified solution. For a fixed task with a binary success reward and independent, identically distributed attempts, let $p > 0$ be success probability and $\bar{c}$ finite mean attempt cost. The standard identity

$$\mathbb{E}C_{\text{until success}} = \frac{\bar{c}}{p} \tag{52}$$

follows by summing $\bar{c}(1-p)^{i-1}$ over attempts $i \geq 1$; cost and success may be dependent within an attempt. If Proposition 2 applies to that task and $V(\pi_{\text{ref}}) + \Delta - B_{\text{AR}} > 0$, it supplies a lower bound on $p = V(\pi)$. Equation (51) separately defines the per-attempt computation. This connects task accuracy with computation once both are measured; the current budget sweeps report accuracy at specified latent step counts.

The theory suggests evaluating composition residuals at student anchors, teacher error at finer intervals, and paired fresh-continuation verifier rewards. A contextual log-score diagnostic on weighted student tokens is

$$\Delta_{\text{ctx}} = \frac{\sum_{(q,n,m,j)\in\mathcal{I}} w_{nm}^{\text{text}} [\log q_{nmj}^{+}(y_{nmj}) - \log p_{nmj}(y_{nmj})]}{\sum_{(q,n,m,j)\in\mathcal{I}} w_{nm}^{\text{text}}}. \tag{53}$$

It measures fit to sampled tokens. For an unweighted token drawn from p at a fixed prefix, its expected log-score difference is $-D_{\mathrm{KL}}(p\|Q)$. The task-level teacher advantage in Proposition 2 is instead measured by verifier reward on fresh continuations.

E Additional Experimental Results

We provide additional analysis on the contribution of the generated latent workspace. Table 8 tests whether answer generation benefits from the model's completed workspace. For flow map, replacing that workspace with RMS-scaled Gaussian noise lowers MBPP+ pass@1 from 57.0% to 48.3%, and replacing it with zero lowers the score to 48.8%. Both changes have paired intervals below zero. SFT also benefits from its normal workspace, with drops of 15.0 and 20.5 points under the two replacements. Joint GRPO attains 54.4% with its normal workspace and 58.3%/57.9% under the replacements.

Together, these measurements show that the flow-map checkpoint used in the ablation retains a useful dependence on its generated workspace while improving its normal-workspace accuracy over the initialization and GRPO checkpoint used in this comparison. The contribution tested here is the effect of the completed representation on answer quality at this checkpoint.

Table 8: Workspace-content ablation on MBPP+: pass@1 (%) at fixed checkpoints. All workspaces are first generated with the adopted 30-step ODE protocol. Gaussian noise is scaled by the original workspace RMS, as in Equation (49). Here $\uparrow$ / $\downarrow$ compares a replacement with *the same row's normal workspace*, using paired 95% intervals.

Configuration	Normal	Gaussian replacement	Zero replacement
SFT initialization	42.3	$27.3^{\downarrow}$	$21.8^{\downarrow}$
Joint GRPO	54.4	$58.3^{\uparrow}$	$57.9^{\uparrow}$
Flow map w/o text OPSD	57.0	$48.3^{\downarrow}$	$48.8^{\downarrow}$