LOOP-DISTILL: REFINING INFORMATION FLOW IN DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) reduce decoding latency by denoising multiple token positions in parallel. A practical way to obtain this is to convert a pretrained autoregressive (AR) large language model into a DLM with the block diffusion paradigm. However, this conversion also changes the role of a forward pass. Instead of predicting one next token from a fully observed prefix, the model must form and refine representations for multiple masked positions under partial context. We show that this mismatch gives rise to an information flow bottleneck: masked token states stabilize prematurely in the shallow layers, yet the predictions made by these layers are often overconfident and rarely consistent with the final output. To address this, we propose *Loop-Distill*, which uses local shallow-layer loop refinement to let masked states absorb known context more fully and distills this deeper shallow refinement into the original single-pass model. During training, we exploit DLM decoding as an intrinsic refinement loop by constructing an online stop-gradient teacher from the student's current weights and looping selected shallow layers to refine masked states. This compresses useful loop computation into the deployed model without changing its inference-time architecture. Across 13 benchmarks, Loop-Distill improves average accuracy on GLM-4.7-Flash and Qwen-30B-A3B-2507 by 2.05 and 2.21 points, respectively. It also improves decoding throughput by 11.0% on average by enabling more effective masked state refinement and thus fewer full-model denoising passes.

1 INTRODUCTION

Autoregressive (AR) language models are powerful but decode tokens sequentially, making generation inherently serial. Diffusion language models (DLMs) alleviate this bottleneck by denoising multiple positions in parallel Zhu et al. (2026); Bie et al. (2025; 2026); Zhu et al. (2025a); AI et al. (2026). In particular, block diffusion is recognized as a mainstream paradigm for DLMs, owing to its strong empirical scaling properties. Since it preserves causal dependencies across blocks while jointly denoising within the active block, it also offers a practical pathway for converting pretrained AR models into DLMs Bie et al. (2025; 2026); AI et al. (2026). Prior work has focused on how to make this conversion effective, including changes to attention, training objectives, and representation alignment (Gong et al., 2025; Fu et al., 2025; Peng et al., 2026; Bie et al., 2025; Fu et al., 2026). We study a complementary question: how the representations of masked tokens, a central component of DLMs, evolve along model depth during decoding.

Conversion changes the computational role of a forward pass. In AR modeling, one pass produces a next-token distribution conditioned on the previous token. In block diffusion, by contrast, one pass must simultaneously construct representations for several masked positions under partially observed context, and the network is invoked repeatedly until all positions in the active block are committed. Generation therefore unfolds on two coupled timescales: transformer layers refine token states *within* a pass, while denoising repeatedly reapplies the full model *across* passes to refine positions.

However, our measurements reveal a breakdown in this coupling (Figure 1). In vanilla converted DLMs, masked token representations undergo drastic updates in the shallow layers and then stabilize prematurely, even though the predictions made by these layers remain highly unreliable. Among high-confidence predictions from the first layer, only a small fraction agree with the final layer's top-1 prediction, and many collapse to the same token. This pattern points to a shallow information flow bottleneck: masked token representations begin to stabilize too early and thereafter receive only minor refinements before sufficient contextual evidence has been incorporated.

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

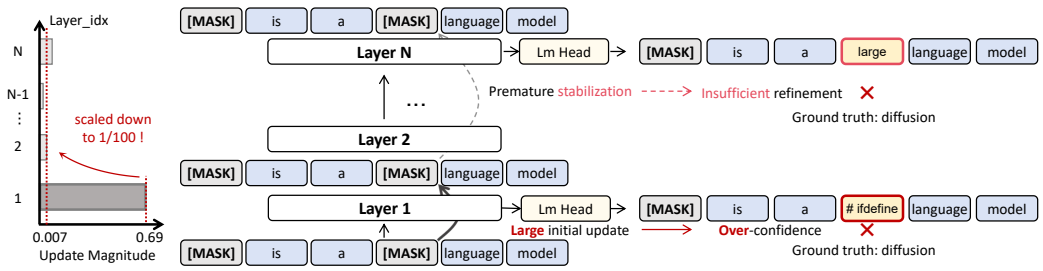


Figure 1: **Information flow bottleneck in converted diffusion language models:** masked token states undergo a large first update followed by weak refinement in subsequent layers, while predictions made by shallow layers are overconfident and collapse to several meaningless words.

This bottleneck is particularly harmful in DLMs because shallow masked token states directly affect masked token predictions during denoising. When these states are insufficiently refined, the model not only makes poorer within-pass predictions but also denoises less efficiently across passes. Motivated by the iterative nature of DLM decoding and the shallow information-flow bottleneck, we propose **Loop-Distill**, a training-only self-distillation method that loops shallow layers so masked states can absorb known context more fully before being distilled back into the single-pass model. Because decoding can itself be viewed as an outer global loop, where the model is repeatedly applied across denoising passes to iteratively refine masked positions, adding a local loop at the problematic shallow layers provides a natural and targeted way to improve shallow masked state refinement.

During training, an online stop-gradient teacher repeatedly applies selected shallow layers to refine masked token states, and the student learns to compress this looped computation into its standard single-pass forward path. The resulting gains mainly come from improving shallow masked state information flow in the deployed model, with the looped trajectory serving as a training signal for that refinement. As a result, Loop-Distill strengthens information flow while leaving inference-time architecture unchanged, avoiding extra deployment overhead and preserving compatibility with existing acceleration stacks. We further analyze shallow masked state refinement from a local refinement perspective, which provides practical guidance for choosing loop placement and loop depth.

Empirically, Loop-Distill enhances information flow, leading to better representation quality and more efficient denoising. Across 13 benchmarks and two block sizes, it improves GLM-4.7-Flash and Qwen-30B-A3B-2507 by 2.05 and 2.21 points on average, respectively. It also improves decoding throughput by 11.0% on average. Contributions are summarized as follows:

- We reveal an information flow bottleneck in DLM: masked token states are updated sharply in shallow layers and become prematurely stable before sufficient context is integrated.
- We introduce **Loop-Distill**, a training-only online distillation method that transfers shallow loop refinement into the original single-pass model without altering model architecture.
- We provide practical loop design guidelines and verify its effectiveness on two large-scale 30B-A3B MoE models across 13 benchmarks under different block sizes.

2 Related Work

2.1 From Autoregressive to Diffusion Language Models

Diffusion language models (DLMs) reduce the serial bottleneck of autoregressive (AR) decoding by predicting and revising multiple masked positions in each denoising step. Among them, block diffusion has emerged as a particularly practical paradigm because it preserves causal dependencies across blocks while enabling parallel refinement within each active block Bie et al. (2025; 2026); Zhu et al. (2025a); AI et al. (2026); Cheng et al. (2026); He et al. (2026). A common way to obtain a strong block-diffusion model is therefore to adapt a pretrained AR model through continued pretraining and supervised fine-tuning, leveraging the capability and scale of existing AR checkpoints.

Prior work on AR-to-block-diffusion adaptation has mainly focused on preserving the capabilities of the source AR model during conversion. One line of work mitigates the attention mismatch caused by bidirectional denoising, either by gradually exposing right context or by preserving block-

structured dependencies (Gong et al., 2025; Fu et al., 2025). A second line of work uses progressive adaptation curricula. For example, LLaDA 2.0 treats AR as block diffusion with block size 1 and gradually increases the block size during training to ease the transition to diffusion-style denoising (Bie et al., 2025). A third line of work uses representation alignment to keep the converted model close to the source AR representation space (Peng et al., 2026). More recently, some works have gone beyond pure conversion and instead adopt joint AR–diffusion training objectives, training diffusion language models together with autoregressive losses to retain AR-compatible behaviors while acquiring diffusion-style decoding capabilities (Yu et al., 2026; Fu et al., 2026).

Our focus is complementary but distinct from these directions. Rather than preserving AR representations through alignment or smoothing the conversion process at the objective or attention level, we study how masked token information is formed and refined within the converted model.

2.2 Recurrent and Looped Language Models

Recurrent-depth and looped models reuse parameters to increase latent computation without proportionally increasing model size. Classic looped transformers repeatedly apply a shared block, while more recent language models use adaptive exits, token-dependent depth, or partially shared middle layers to allocate the loop more effectively (Zhu et al., 2025c; Bae et al., 2025; Zhu et al., 2025b; Chen et al., 2026). However, this line of work is mostly developed for autoregressive models, where each forward pass is defined around next-token prediction rather than refinement, making the loop paradigm a less natural fit. A closely related recent effort, LoopMDM (Lee et al., 2026), introduces a DLM with an internal single-layer loop architecture. However, it is studied primarily at the 100M scale with pure diffusion, rather than in mainstream block diffusion with large-scale parameters.

Loop-Distill differs from these directions along both purpose and deployment. In purpose, loop is used only to construct a training-time teacher from the same block-diffusion model, and only at the shallow layers where masked token information flow is defective. In deployment, Loop-Distill introduces no loop architecture at inference time and leaves the original model unchanged. This training-only design is especially practical because it avoids inference loop overhead and remains compatible with acceleration ecosystems (*e.g.*, SGLang) and optimized low-level CUDA kernels.

3 Preliminaries

Block diffusion combines autoregressive generation across blocks with diffusion-style denoising within each block (Bie et al., 2025). Let $\boldsymbol{x} = (x_1, \ldots, x_N)$ be a prompt-response sequence partitioned into contiguous blocks of size B (**denoted as block size**), and let $b(i) = \lfloor (i-1)/B \rfloor$ denote the block index of position i. Under block attention, position i attends to all positions in preceding blocks and to all positions in its own block:

$$A_{ij} = 1 \quad \Longleftrightarrow \quad b(j) \le b(i). \tag{1}$$

For supervised fine-tuning, response tokens are partially masked while the prompt and previously completed blocks remain visible. Let $\tilde{\boldsymbol{x}}$ denote the corrupted sequence and $\mathcal{M} = \{i : \tilde{x}_i = \texttt{[MASK]}\}$ the masked response positions. The training objective is masked token cross-entropy:

$$\mathcal{L}_{\mathrm{BD}}(\theta) = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log p_\theta(x_i \mid \tilde{\boldsymbol{x}}; A). \tag{2}$$

At inference, the active block is initialized with B masks and refined over multiple denoising steps. At step s, the model processes $\boldsymbol{x}^{(s)}$ and assigns each remaining masked position $i \in \mathcal{M}^{(s)}$ a candidate token $\hat{x}_i^{(s)}$ with confidence $c_i^{(s)} = p_\theta(\hat{x}_i^{(s)} \mid \boldsymbol{x}^{(s)}; A)$. Given a decoding threshold τ, positions whose confidence exceeds the threshold are committed: $\mathcal{C}^{(s)} = \{\, i \in \mathcal{M}^{(s)} : c_i^{(s)} \ge \tau \,\}$. If $\mathcal{C}^{(s)} = \varnothing$, a fallback commits the highest-confidence remaining position. The sequence is then updated as

$$x_i^{(s+1)} = \begin{cases} \hat{x}_i^{(s)}, & i \in \mathcal{C}^{(s)}, \\ x_i^{(s)}, & i \notin \mathcal{C}^{(s)}. \end{cases} \tag{3}$$

Committed tokens become visible context for the next step, and decoding continues until no masked positions remain. Block diffusion is therefore an *outer refinement loop*: the same full model is repeatedly applied to an unresolved block.

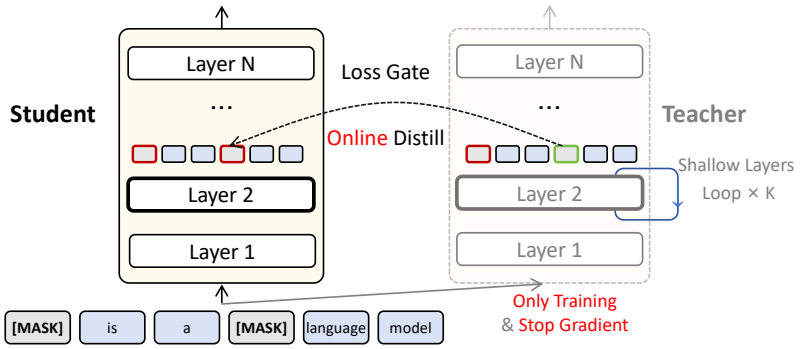


Figure 2: **Loop-Distill.** During training, an *online shared-parameter stop-gradient teacher* loops selected shallow layers K times to refine masked token states. Distillation is applied only at masked positions and only when the looped teacher achieves lower masked token loss than the student.

4 Loop-Distill

4.1 Shallow-Layer Local Refinement

Let the transformer consist of L residual layers, $F_{1:L} = f_L \circ \cdots \circ f_1$. Given a corrupted input sequence, the deployed model computes hidden states through the standard layer-by-layer forward pass:

$$h^{(0)} = \mathrm{Embed}(\tilde{\boldsymbol{x}}), \qquad h^{(\ell)} = f_\ell(h^{(\ell-1)}), \quad \ell = 1, \ldots, L. \tag{4}$$

For masked positions, this trajectory must convert a shared mask embedding into a context-specific representation. Our analysis shows that the main defect lies in shallow layers: the first update is overly large, while subsequent shallow layers change little despite unstable token preferences.

Rather than seeking additional refinement through another full outer-loop pass, we refine only the problematic shallow layers. Reapplying a selected layer yields a local refinement trajectory:

$$h \to f_\ell(h) \to f_\ell^2(h) \to \cdots \to f_\ell^K(h), \tag{5}$$

where f_ℓ^K denotes K repeated applications of the same layer. The resulting endpoint provides a targeted teacher signal for improving shallow masked state refinement.

4.2 Loss-Gated masked state Distillation

We realize this supervision with an online teacher branch constructed from the same model at each training step. The teacher and student process the same corrupted example and share parameters, but differ in computation: the teacher performs shallow local looping and is detached by stop-gradient.

Let $\mathcal{S} \subset \{1, \ldots, L\}$ denote the shallow layers selected for looping, let $\ell_0 = \min \mathcal{S}$, and let q be a distribution over candidate loop counts. For a sampled $K \sim q$, the student follows the original single-pass path in Equation 4, while the online stop-gradient teacher reuses the selected shallow layers K times to build a locally refined trajectory. Figure 2 illustrates the overall training pipeline. The teacher starts from the student's pre-loop hidden state,

$$\tilde{h}^{(\ell_0-1,K)} = \mathrm{sg}\Big(h^{(\ell_0-1)}\Big), \tag{6}$$

and then propagates as

$$\tilde{h}^{(\ell,K)} = \begin{cases} f_\ell^K(\tilde{h}^{(\ell-1,K)}), & \ell \in \mathcal{S}, \\ f_\ell(\tilde{h}^{(\ell-1,K)}), & \ell \notin \mathcal{S}, \end{cases} \qquad \ell = \ell_0, \ldots, L. \tag{7}$$

Here $\mathrm{sg}(\cdot)$ stops gradients through the teacher branch, so the teacher acts only as an online target. Also, f_ℓ^K denotes K repeated applications of layer ℓ. Only the selected shallow layers are looped, while all remaining layers are applied once.

For every loop layer $\ell \in \mathcal{S}$, we distill the student hidden state $h^{(\ell)}$ toward the corresponding teacher hidden state $\tilde{h}^{(\ell,K)}$, but only at masked positions $i \in \mathcal{M}$:

$$d_{\ell,i}^{(K)} = 1 - \frac{\langle h_i^{(\ell)}, \mathrm{sg}(\tilde{h}_i^{(\ell,K)}) \rangle}{\|h_i^{(\ell)}\|_2 \, \|\tilde{h}_i^{(\ell,K)}\|_2 + \epsilon}, \qquad \mathcal{L}_{\mathrm{traj}}^{(K)} = \frac{1}{|\mathcal{S}| \, |\mathcal{M}|} \sum_{\ell \in \mathcal{S}} \sum_{i \in \mathcal{M}} d_{\ell,i}^{(K)}. \tag{8}$$

Here $d_{\ell,i}^{(K)}$ is the cosine distance between student and teacher states at layer ℓ and position i, $\langle\cdot,\cdot\rangle$ and $\|\cdot\|_2$ denote the Euclidean inner product and norm, and $\epsilon > 0$ prevents division by zero. The normalization by $|\mathcal{S}|$ and $|\mathcal{M}|$ averages the loss over looped layers and masked positions.

Restricting distillation to the masked set $\mathcal{M}$ is aligned with both the task and the diagnosed failure mode. The positions in $\mathcal{M}$ are exactly those reconstructed by the block-diffusion objective in Equation 2, and the masked token does not appear in AR next-token pretraining. Their hidden states therefore require direct adaptation, whereas visible prompt tokens and already completed response tokens primarily serve as conditioning context rather than prediction targets.

The trajectory loss in Equation 8 encourages the student to match the teacher's masked state refinement path. However, a looped teacher path is useful only if its additional local computation also leads to a better task-level prediction. We therefore apply distillation only when the looped teacher improves the task objective on the same masked reconstruction set. To make this comparison, we score the teacher with the same masked token cross entropy as in Equation 2:

$$\tilde{\mathcal{L}}_{\mathrm{BD}}^{(K)} = -\frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}} \log \operatorname{softmax}\Big(W_{\mathrm{out}}\,\mathrm{Norm}(\tilde{h}_i^{(L,K)})\Big)_{x_i}, \tag{9}$$

where W_{out} is the shared output projection, Norm is the final normalization, and the subscript x_i selects the probability assigned to the ground-truth token x_i. We then keep the teacher signal only if it improves over the student's masked reconstruction loss:

$$g_K = \mathbb{I}\Big[\tilde{\mathcal{L}}_{\mathrm{BD}}^{(K)} < r\,\mathcal{L}_{\mathrm{BD}}\Big], \qquad r = 1.0, \tag{10}$$

where $\mathbb{I}[\cdot]$ is the indicator function and r is the required teacher–student loss ratio. In the main configuration, $r = 1.0$, so $g_K = 1$ exactly when the teacher achieves lower masked token cross entropy than the student on the same batch and masked positions; otherwise, $g_K = 0$ and no distillation signal is applied. Therefore, the resulting Loop-Distill objective is

$$\mathcal{L}_{\mathrm{LoopDistill}} = \mathcal{L}_{\mathrm{BD}} + \lambda\,\mathbb{E}_{K\sim q}\left[g_K\,\mathcal{L}_{\mathrm{traj}}^{(K)}\right], \qquad \lambda = 1.0, \tag{11}$$

where λ controls the weight of the auxiliary trajectory term. Sampling $K \sim q$ turns the auxiliary term in Equation 11 into an expectation over a family of teacher trajectories, rather than supervision from a single fixed loop depth. This is useful because the amount of beneficial local refinement can vary across examples and training stages. Equations 7–11 specify how the looped teacher is used, but not how its layers or depth are chosen; we address these choices next through the local refinement dynamics of masked token states.

4.3 Looped Teacher Design from a Local Refinement Perspective

We design the looped teacher by analyzing the local refinement dynamics of masked token states. Let $h_{\mathcal{M}}^{(\ell)} \in \mathbb{R}^{|\mathcal{M}|\times d}$ denote the hidden states at masked positions after layer ℓ, where $\mathcal{M}$ is the masked set and d is the hidden dimension. For each layer, the masked token transition is written as

$$h_{\mathcal{M}}^{(\ell+1)} = T_\ell\Big(h_{\mathcal{M}}^{(\ell)}; c^{(\ell)}\Big), \tag{12}$$

where T_ℓ is the effective update induced by layer ℓ, and $c^{(\ell)}$ denotes the contextual states visible at that layer, including prompt tokens, decoded history, and other positions in the active block. Reapplying a layer or short shallow block defines a local refinement trajectory on masked positions, which we use to determine three design choices: the *loop start*, the *loop span*, and the *loop depth*.

Because the information flow bottleneck identified above is concentrated in the shallow layers, we place the loop as early as possible among the layers that already perform contextual masked token refinement. In the MoE backbones studied here, this corresponds to the shallowest MoE layer, which is also a natural loop point because routing can provide different expert-specialized updates across loop iterations. With the loop start fixed in this way, we first determine the maximum useful loop count $K_{\max}$ and then choose the loop span under a drift constraint.

How many loop steps are needed? We first ask how many times a fixed local loop operator should be reapplied. Let T denote the refinement operator induced by one loop step, which may be a single layer or several layers. Starting from the pre-loop masked states, repeated application of T produces

$$h_{\mathcal{M}}^{(k+1)} = T\Big(h_{\mathcal{M}}^{(k)}; c\Big), \qquad k = 0, 1, 2, \ldots, \tag{13}$$

with update magnitudes

$$\Delta_k := \left\| h_{\mathcal{M}}^{(k+1)} - h_{\mathcal{M}}^{(k)} \right\|_2 . \tag{14}$$

If the loop acts as a stable local refinement process, the masked states should absorb useful contextual information quickly, so the incremental updates should diminish after the first few iterations. We therefore normalize later updates by the first one and define the relative gain ratio

$$R_k := \frac{\Delta_k}{\Delta_0 + \epsilon}, \qquad k \geq 1, \tag{15}$$

where $\epsilon > 0$ prevents division by zero. This ratio measures the incremental gain of each additional loop step relative to the first. We increase the loop count until the additional gain becomes negligible:

$$R_k \leq \eta. \tag{16}$$

Let $K_{\max}$ denote the smallest k satisfying Equation 16, which we use as the maximum useful loop depth for teacher construction.

How many layers should be looped? We next ask how large the looped shallow block should be under this effective refinement scale. Expanding the loop span can strengthen the teacher, but it also increases the cumulative deviation from the student's single-pass trajectory. This trade-off matters because the task-level loss gate in Equation 10 already filters the teacher signal: a stronger teacher may improve masked token reconstruction, but an overly aggressive one can drift too far from the deployed model's native path and therefore activate the gate less often.

We therefore choose the loop span by controlling the cumulative masked token drift induced by the loop. Let the looped layers form a contiguous segment $\mathcal{S}_m = \{\ell_0, \ell_0+1, \ldots, \ell_0+m-1\}$, where ℓ_0 is the starting layer and m is the number of consecutive looped layers. For the effective loop depth K, we measure the cumulative teacher–student drift on masked positions over this segment by

$$D_m^{(K)} := \frac{1}{|\mathcal{S}_m|\,|\mathcal{M}|} \sum_{\ell \in \mathcal{S}_m} \sum_{i \in \mathcal{M}} \left(1 - \frac{\langle h_i^{(\ell)}, \mathrm{sg}(\tilde{h}_i^{(\ell,K)}) \rangle}{\|h_i^{(\ell)}\|_2 \, \|\tilde{h}_i^{(\ell,K)}\|_2 + \epsilon} \right), \tag{17}$$

the average cosine-distance drift between the student's shallow states and the looped teacher on the masked set. We then choose the smallest shallow span whose cumulative drift remains within a fixed budget:

$$D_m^{(K)} \leq \delta. \tag{18}$$

Together with the loss gate in Equation 10, this favors the lightest loop span that is still useful: strong enough to improve masked reconstruction, but not so aggressive that excessive drift suppresses the teacher signal on many informative examples.

Default loop configuration setup. For each backbone, we determine the loop *before training* by starting from the earliest layer that performs contextual masked token refinement, setting $K_{\max}$ using Equation 16, and expanding the contiguous shallow span only while the masked token drift in Equation 18 stays within budget. This gives $\mathcal{S} = \{2, 3\}$ for GLM-4.7-Flash and $\mathcal{S} = \{1, 2, 3\}$ for Qwen-30B-A3B-2507, with $K_{\max} = 3$ in both cases. Section 5.3 verifies these default settings.

5 Experiments

5.1 Experimental Setup

Models and training. We evaluate Loop-Distill on two instruction-tuned 30B-A3B MoE backbones converted to block diffusion: GLM-4.7-Flash and Qwen-30B-A3B-2507. Training uses supervised fine-tuning only on the full Step-3.5-Flash SFT set Huang et al. (2026) (about 1.62M examples covering reasoning, coding, and general instruction following). The auxiliary loss weight is $\lambda = 1.0$. We train all models with Megatron-LM Shoeybi et al. (2020) distributed training on 128 NVIDIA B200 GPUs. Both models use the same optimization setup: sequence length 16,384, global batch size 128, 1200 training steps, and the Muon optimizer (Liu et al., 2025) in bfloat16 precision.

Online teacher. Loop-Distill uses an online stop-gradient teacher instantiated from the student's current weights at each training step, rather than a trained offline teacher. This requires no additional pretraining and keeps the teacher aligned with the student's current representation space.

Table 1: **Main results.** We report accuracy under four settings (bs: block size); compared with the same AR-to-BD converted model trained with standard block-diffusion SFT only (Vanilla), Loop-Distill (LD) consistently improves average performance on 13 benchmarks.

		GLM, bs=32			Qwen, bs=32			GLM, bs=4			Qwen, bs=4		
Category	Benchmark	Vanilla	LD	Δ	Vanilla	LD	Δ	Vanilla	LD	Δ	Vanilla	LD	Δ
	Ave.	46.68	**48.84**	*+2.17*	46.04	**48.37**	*+2.33*	56.31	**58.24**	*+1.93*	56.53	**58.62**	*+2.09*
General	IFBench	32.40	**36.20**	*+3.80*	28.13	**28.80**	*+0.67*	41.27	**43.20**	*+1.93*	31.07	**32.40**	*+1.33*
	KorBench	50.32	**50.72**	*+0.40*	**42.40**	42.24	*-0.16*	52.96	**56.00**	*+3.04*	55.16	**56.64**	*+1.48*
	MMLU-Pro	46.93	**50.05**	*+3.12*	**49.43**	48.54	*-0.89*	57.48	**59.98**	*+2.50*	57.92	**60.12**	*+2.20*
Code	MBPP+	65.31	**69.87**	*+4.56*	**66.30**	66.24	*-0.06*	74.61	**76.82**	*+2.21*	75.93	**76.85**	*+0.92*
	HumanEval+	67.15	**68.37**	*+1.22*	65.24	**71.49**	*+6.25*	77.55	**79.65**	*+2.10*	79.57	**83.38**	*+3.81*
	BigCodeBench	22.72	**26.48**	*+3.76*	25.44	**27.28**	*+1.84*	34.65	**36.32**	*+1.67*	34.47	**38.68**	*+4.21*
	LiveCodeBench	20.10	**21.32**	*+1.22*	19.88	**23.29**	*+3.41*	26.30	**27.42**	*+1.12*	24.39	**26.87**	*+2.48*
Math	MATH	**76.84**	75.88	*-0.96*	76.98	**77.48**	*+0.50*	87.72	**88.48**	*+0.76*	87.40	**88.62**	*+1.22*
	OlympiadBench	49.08	**50.52**	*+1.44*	44.41	**49.85**	*+5.44*	54.41	**55.63**	*+1.22*	54.26	**55.52**	*+1.26*
Reasoning	GPQA-Diamond	42.39	**43.69**	*+1.30*	40.69	**44.76**	*+4.07*	45.96	**48.67**	*+2.71*	44.41	**45.04**	*+0.63*
	BBH	48.88	**52.80**	*+3.92*	57.82	**58.80**	*+0.98*	70.07	**71.34**	*+1.27*	74.20	**75.34**	*+1.14*
	MuSR	59.19	**61.95**	*+2.76*	58.55	**61.87**	*+3.32*	70.38	**71.39**	*+1.01*	72.16	**72.84**	*+0.68*
	ZebraLogic	25.47	**27.12**	*+1.65*	23.25	**28.23**	*+4.98*	38.68	**42.27**	*+3.59*	43.97	**49.78**	*+5.81*

Looped teacher configuration. We use the default loop setup from Section 4.3 with $\delta = 0.15$ and $\eta = 0.05$. This yields $\mathcal{S} = \{2, 3\}$ for GLM-4.7-Flash and $\mathcal{S} = \{1, 2, 3\}$ for Qwen-30B-A3B-2507. For GLM-4.7-Flash, layer 1 is a dense layer rather than an MoE layer, so the earliest loop start selected is layer 2. We sample K uniformly from $\{2, 3\}$. More details are provided in Appendix E.

Evaluation. We evaluate on 13 benchmarks spanning general and reasoning (IFBench Pyatkin et al. (2026), KorBench Ma et al. (2025), GPQA-Diamond Rein et al. (2023), BBH Suzgun et al. (2023), MMLU-Pro Wang et al. (2024), ZebraLogic Lin et al. (2025), MuSR Sprague et al. (2024)), code generation (MBPP+ Austin et al. (2021), HumanEval+ Liu et al. (2023), BigCodeBench Zhuo et al. (2025), LiveCodeBench v6 Jain et al. (2025)), and mathematical or logical reasoning (MATH Hendrycks et al. (2021), OlympiadBench He et al. (2024)). All benchmarks are evaluated with 4 random samples, and we report the mean pass@1. Throughput is measured as generated tokens per second (TPS) with batch size 1 using SGLang on 8×H20-3E GPUs.

5.2 Main Results

Table 1 shows that Loop-Distill consistently improves performance across all four model settings. For block size 32, it raises the overall 13-benchmark mean by 2.17 points on GLM-4.7-Flash and by 2.33 points on Qwen-30B-A3B-2507. For block size 4, it further improves the mean by 1.93 points on GLM-4.7-Flash and by 2.09 points on Qwen-30B-A3B-2507. The gains are broad rather than domain-specific, but they are especially pronounced on coding and reasoning benchmarks. This pattern holds under both block sizes: at block size 32, the largest gains appear on tasks such as MBPP+, BigCodeBench, BBH, HumanEval+, and GPQA-Diamond, while at block size 4 the greatest improvements remain concentrated on reasoning- and code-intensive benchmarks such as ZebraLogic, BigCodeBench, and HumanEval+. This consistency across backbones and block sizes supports our claim that Loop-Distill addresses a general weakness of AR-to-block-diffusion conversion rather than a backbone-specific or block-size-specific artifact. Loop-Distill is also complementary to the AR-representation alignment objective; combining the two yields further gains (Appendix C). Unless otherwise noted, the remaining analyses are conducted on GLM-4.7-Flash with block size 32.

5.3 Ablation Studies

Strategy ablations. Table 2 isolates the main components of Loop-Distill. Removing the loss gate hurts performance, showing that an ungated loop introduces harmful supervision on a nontrivial fraction of examples. Distilling all tokens is also weaker than masked-only distillation, consistent with our diagnosis that the bottleneck lies in masked token state formation rather than visible-context representations. Replacing trajectory distillation with final-state-only supervision underperforms the full method, suggesting that shaping the intermediate shallow representation refinement path matters beyond matching a better endpoint. Finally, sampling K slightly outperforms a fixed $K = 3$, indicating that exposure to a small family of useful loop trajectories improves robustness.

Table 2: **Strategy ablations.** The average is over GPQA-Diamond, MMLU-Pro, LiveCodeBench-v6, and Olympiad-Bench. The full configuration performs best overall.

Method	*Ave.*	GPQA-D	MMLU-Pro	LCB-v6	Olympiad
Vanilla	39.63	42.39	46.93	20.10	49.08
Loop-Distill	**41.40**	**43.69**	**50.05**	21.32	**50.52**
w/o loss gate	40.31	42.91	48.34	20.88	49.10
all tokens	40.58	43.02	48.90	21.01	49.38
final-state	40.74	43.18	49.11	21.06	49.62
fixed $K = 3$	41.07	43.51	49.56	**21.44**	49.78

Table 3: **Loss gate ablation.** Each column uses the teacher–student loss comparison $\tilde{\mathcal{L}}_{\mathrm{BD}} < r\mathcal{L}_{\mathrm{BD}}$. The balanced gate performs best.

r value	0.95	1.00	1.05
Average	40.03	**41.40**	40.22
GPQA-D	42.39	**43.69**	41.67
MMLU-Pro	47.60	**50.05**	47.64
LCB-v6	20.26	21.32	**22.14**
Olympiad	49.85	**50.52**	49.44

Table 4: **Looped teacher design ablations.** Average performance is reported over GPQA-Diamond, MMLU-Pro, LiveCodeBench-v6, and OlympiadBench. The chosen configuration gives the best overall trade-off. GA rate denotes the average gate activation rate during training.

Factor	Setting	Acc	GA rate (%)	Diagnostic statistic	Analysis
Loop start	layer 2 (shallow start)	41.40	–	–	best placement
	layer 2/N-1 (middle start)	40.51	–	–	refine too late
	layer N-2 (deep start)	40.08	–	–	far from bottleneck
Loop span	1 layer ($\delta = 0.05$)	41.01	27.0	$D_m^{(K)} = 0.04$	under-refined
	2 layers (default $\delta = 0.15$)	41.40	23.7	$D_m^{(K)} = 0.13$	best drift/benefit
	3 layers ($\delta = 0.40$)	40.36	18.8	$D_m^{(K)} = 0.38$	excess drift
Loop depth	$K_{\max} = 2$ ($\eta = 0.40$)	40.27	–	$R_2 = 0.31$	stops too early
	$K_{\max} = 3$ (default $\eta = 0.05$)	41.40	–	$R_3 = 0.04$	near elbow
	$K_{\max} = 4$ ($\eta = 0.02$)	41.46	–	$R_4 = 0.01$	marginal gain

Loss gate ablation. Table 3 studies the gate threshold in Equation 10. The balanced setting $r = 1.00$ gives the best mean performance. A stricter gate ($r = 0.95$) suppresses useful teacher signals, while a looser one ($r = 1.05$) admits more noisy trajectories. This supports the design choice of distilling only when the looped teacher is actually better on the same masked reconstruction task.

Looped teacher design ablation. Table 4 directly validates the design rules in Section 4.3. First, the best results are obtained when the loop starts from the earliest shallow layer that already performs contextual masked token refinement, consistent with the guideline of placing the loop as early as possible. Second, loop span is best chosen under the masked state drift budget: a 2-layer shallow loop gives the strongest average accuracy while keeping the cumulative drift within the target budget, whereas expanding the span further causes excessive drift and lower gate usage. Third, loop depth follows the relative-gain criterion: performance improves from $K_{\max} = 2$ to $K_{\max} = 3$, but the additional refinement gain is already near saturation at $K_{\max} = 3$, with only a marginal further drop at $K_{\max} = 4$. Overall, the default configuration matches the intended guideline-based trade-off.

5.4 Training Dynamics and Decoding Behavior

We further analyze Loop-Distill from four complementary views: gate activation dynamics, decoding efficiency, shallow-layer information flow, and premature token commitments. The decoding analyses are reported on GPQA-Diamond, MMLU-Pro, LiveCodeBench-v6, and OlympiadBench.

Gate activation dynamics. Figure 3 plots gate activation over training. The activation rate decreases monotonically, which is the expected signature of successful self-distillation: early in training, the looped teacher often achieves lower masked reconstruction loss than the student, whereas later the student progressively absorbs this refinement behavior and becomes less dependent on the teacher.

Decoding efficiency. Loop-Distill also improves the outer block-diffusion decoding process. As shown in Table 5, it consistently increases the average number of decoded tokens per forward pass (TPF) under both confidence thresholds, improving the average TPF by 4.43% at threshold 0.90 and by 6.69% at threshold 0.95. This indicates that Loop-Distill teaches the model to make more effective shallow refinements, so more masked positions are resolved in each pass, and fewer full-model denoising passes are needed. These decoding improvements translate into higher measured throughput in Table 6. Loop-Distill improves TPS in all four settings, with relative gains of 12.2%, 11.3%, 10.1%, and 10.5%, respectively. Detailed per-benchmark throughput is provided in Appendix A.

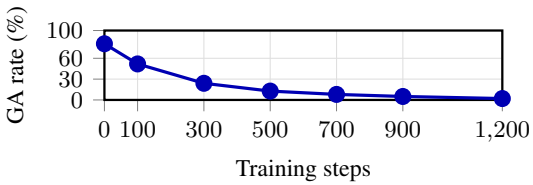


Figure 3: **Gate activation (GA) rate over training.** The fraction of batches satisfying $\tilde{\mathcal{L}}_{\mathrm{BD}}^{(K)} < \mathcal{L}_{\mathrm{BD}}$ decreases steadily as the student internalizes the shallow looped teacher.

Table 5: **Decoding efficiency.** We report the average number of decoded tokens per forward pass (TPF). Loop-Distill consistently improves TPF.

Threshold	Benchmark	Vanilla	Loop-Distill	Gain
0.90	GPQA-D	2.887	**2.999**	+3.88%
	Olympiad	1.903	**2.136**	+12.27%
	LCB-v6	1.963	**2.048**	+4.34%
	MMLU-Pro	3.029	**3.063**	+1.11%
0.95	GPQA-D	2.876	**3.063**	+6.49%
	Olympiad	1.990	**2.251**	+13.08%
	LCB-v6	1.841	**1.969**	+6.96%
	MMLU-Pro	2.974	**3.014**	+1.36%

Table 6: **Decoding throughput comparison.** TPS (tokens per second) is averaged over 9 benchmarks; bs indicates block size. Loop-Distill consistently improves average throughput in all backbones and block sizes.

Setting	Vanilla	Loop-Distill	Gain
GLM, bs 32	491.91	**551.86**	+12.2%
Qwen, bs 32	266.41	**296.61**	+11.3%
GLM, bs 4	276.29	**304.14**	+10.1%
Qwen, bs 4	174.89	**193.29**	+10.5%

Table 7: **Shallow masked token information flow.** Lower cosine or higher update magnitude indicates a larger refinement; T_i indicates transition i. Compared with Vanilla, Loop-Distill exhibits stronger later-layer refinement.

Metric	Vanilla	Loop-Distill
T_0 cosine similarity S_0	0.02	0.09
T_1 cosine similarity S_1	0.74	**0.39**
T_0 update magnitude U_0	69.31%	33.08%
T_1 update magnitude U_1	0.74%	**7.25%**

Shallow-layer information flow. To understand why decoding becomes more efficient, we analyze hidden-state refinement for tokens that are eventually committed during diffusion. Let $h_i^{(\ell)}$ denote the hidden state at layer ℓ. We measure adjacent-layer similarity as $S_\ell = \mathbb{E}_i[\cos(h_i^{(\ell)}, h_i^{(\ell+1)})]$ and relative update magnitude as $U_\ell = \mathbb{E}_i\left[\|h_i^{(\ell+1)} - h_i^{(\ell)}\|_2 / (\|h_i^{(\ell)}\|_2 + \epsilon)\right]$. As shown in Table 7, Loop-Distill redistributes shallow layer refinement: it suppresses the overly sharp first update and shifts more refinement to the following transition. This suggests that Loop-Distill delays refinement until more contextual evidence has been incorporated. Appendix D further provides the full curves.

Mechanistically, this refined shallow information flow appears in two complementary parts of the same transformer layer. On the attention side, shallow masked tokens anchor more strongly to reliable decoded context and then engage in richer within-block interaction, indicating improved information acquisition. On the MLP side (with MoE), they exhibit more stable expert assignments in shallow layers, suggesting that shallow masked tokens are more consistently routed to a specialized subset of experts for interpreting partially specified semantics. Together, these results suggest that Loop-Distill improves information flow across a transformer layer, affecting both where masked tokens attend and how they are subsequently routed and processed by experts (Appendix B.1, B.2).

Premature token commitments. We probe intermediate token preferences by applying the shared LM head at layer 1 and comparing top-1 predictions with the final output. In Vanilla, 56.39% of positions exceed 0.8 confidence, but only 18.98% agree with the final prediction, with many collapsing to the same single-space token. Loop-Distill sharply suppresses this shallow overconfidence: only 0.03% of layer-1 positions exceed 0.8 confidence. These results suggest that Loop-Distill prevents shallow layers from forming spurious high-confidence token commitments before sufficient context has been integrated. Appendix B.3 provides token-level evidence for this collapse behavior.

6 Conclusion

We identify the information flow bottleneck in AR-to-DLM conversion: masked token states receive an overly large first update while stabilizing too early. To mitigate this failure mode, we propose **Loop-Distill**, a training-only self-distillation method that uses an online stop-gradient *looped* teacher to supervise masked state refinement in the single-pass model. Across two 30B-A3B MoE backbones and 13 benchmarks, Loop-Distill improves average accuracy by 2.05 points on GLM-4.7-Flash and 2.21 points on Qwen-30B-A3B-2507, while increasing decoding throughput by 11.0%. Quantitative analyses show that the gains mainly come from improving shallow information flow refinement in the single-pass: Loop-Distill reduces shallow premature overconfidence, improves subsequent masked state refinement, and thereby reduces costly full-model iterative decoding.

AI USE STATEMENT

Large language models (LLMs) were used as a writing aid in the preparation of this manuscript. Their use was strictly limited to improving grammar, phrasing, and formatting. All scientific content, including research ideas, methodology, data analysis, and conclusions, was conceived and developed exclusively by the authors. The authors have reviewed all AI-generated suggestions and take full responsibility for the final manuscript's accuracy and integrity.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we will publicly release the source code and the trained model weights upon acceptance of this paper. The release will also include detailed instructions and evaluation to enable other researchers to reproduce our results.

REFERENCES

Inclusion AI, Tiwei Bie, Haoxing Chen, Tieyuan Chen, Zhenglin Cheng, Long Cui, Kai Gan, Zhicheng Huang, Zhenzhong Lan, Haoquan Li, et al. Llada2. 0-uni: Unifying multimodal understanding and generation with diffusion large language model. *arXiv preprint arXiv:2604.20796*, 2026.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. In *International Conference on Learning Representations*, volume 2025, pp. 34282–34327, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

Tieyuan Chen, Xiaodong Chen, Haoxing Chen, Zhenzhong Lan, Weiyao Lin, and Jianguo Li. Dnd: Boosting large language models with dynamic nested depth. In *International Conference on Learning Representations*, volume 2026, pp. 28088–28105, 2026.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, et al. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*, 2025.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Zhihao He, Tieyuan Chen, Kangyu Wang, Ziran Qin, Yang Shao, Chaofan Gan, Shijie Li, Zuxuan Wu, and Weiyao Lin. Vidlada: Bidirectional diffusion large language models for efficient video understanding. *arXiv preprint arXiv:2601.17868*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Ailin Huang, Ang Li, Aobo Kong, Bin Wang, Binxing Jiao, Bo Dong, Bojun Wang, Boyu Chen, Brian Li, Buyun Ma, et al. Step 3.5 flash: Open frontier-level intelligence with 11b active parameters. *arXiv preprint arXiv:2602.10604*, 2026.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Sanghyun Lee, Chunsan Hong, Seungryong Kim, Jonghyun Lee, Jongho Park, and Dongmin Park. Looped diffusion language models, 2026.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. In *International Conference on Learning Representations*, volume 2025, pp. 80062–80161, 2025.

Fred Zhangzhi Peng, Alexis Fox, Anru R Zhang, and Alexander Tong. Don't retrain, align: Adapting autoregressive lms to diffusion lms via representation alignment. *arXiv preprint arXiv:2605.06885*, 2026.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14670–14728, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Yifan Yu, Yuqing Jian, Junxiong Wang, Zhongzhu Zhou, Donglin Zhuang, Xinyu Fang, Sri Yanamandra, Xiaoxia Wu, Qingyang Wu, Shuaiwen Leon Song, et al. Introspective diffusion language models. *arXiv preprint arXiv:2604.11035*, 2026.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, et al. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025a.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025b.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025c.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pp. 66602–66656, 2025.

Table 8: **Per-benchmark measured decoding throughput.** We report generated tokens per second (TPS) under the same decoding configuration for all four model settings. Consistent with the main-text averages, Loop-Distill improves throughput on most benchmark–setting pairs and yields double-digit average TPS gains across all four settings.

Setting	Benchmark	Vanilla	Loop-Distill
GLM-4.7-Flash, block size 32	IFBench	349.93	**381.48**
	KorBench	40.63	**75.43**
	GPQA-Diamond	1005.49	**1276.91**
	MMLU-Pro	353.67	**374.78**
	MuSR	418.77	**457.66**
	MBPP+	252.18	**283.06**
	HumanEval+	**111.66**	101.22
	BigCodeBench	905.84	**1098.36**
	MATH	**1385.59**	1378.99
	(Ave., +12.2%)	491.91	**551.86**
Qwen-30B-A3B-2507, block size 32	IFBench	270.58	**286.92**
	KorBench	31.35	**33.66**
	GPQA-Diamond	449.34	**505.83**
	MMLU-Pro	160.04	**171.94**
	MuSR	358.52	**385.54**
	MBPP+	586.54	**659.59**
	HumanEval+	71.24	**94.57**
	BigCodeBench	126.26	**133.48**
	MATH	598.14	**666.04**
	(Ave., +11.3%)	266.41	**296.61**
GLM-4.7-Flash, block size 4	IFBench	158.85	**172.02**
	KorBench	14.65	**20.70**
	GPQA-Diamond	631.65	**791.45**
	MMLU-Pro	211.87	**247.47**
	MuSR	382.45	**396.55**
	MBPP+	168.79	**206.52**
	HumanEval+	61.15	**83.08**
	BigCodeBench	114.65	**142.11**
	MATH	**1008.95**	962.38
	(Ave., +10.1%)	276.29	**304.14**
Qwen-30B-A3B-2507, block size 4	IFBench	113.21	**125.15**
	KorBench	9.64	**12.41**
	GPQA-Diamond	**613.52**	584.44
	MMLU-Pro	167.15	**210.98**
	MuSR	215.04	**286.02**
	MBPP+	109.76	**133.03**
	HumanEval+	40.94	**41.49**
	BigCodeBench	33.99	**38.33**
	MATH	440.39	**494.38**
	(Ave., +10.5%)	174.89	**193.29**

A PER-BENCHMARK THROUGHPUT RESULTS

Table 8 reports per-benchmark measured decoding throughput for all four model settings. These results complement the model-setting-wise averages in Table 6 in the main text and illustrate how the throughput gains of Loop-Distill vary across tasks.

Consistent with the averages in the main text, Loop-Distill improves throughput on the majority of benchmark–setting pairs, although a small number of cases show neutral or slightly negative changes. Averaged within each model setting, the TPS gains are 12.2%, 11.3%, 10.1%, and 10.5%, corresponding to an overall mean throughput improvement of 11.0% across the four settings.

Table 9: **Attention-side evidence for shallow information flow refinement.** Loop-Distill reorganizes shallow layers' attention over masked tokens: shallow layers rely more on reliable decoded context, while subsequent shallow layers exchange more information within the active block. The higher mask-only attention entropy further indicates less premature attention collapse.

Metric	Vanilla	Loop-Distill
Layer 1: [MASK] $\rightarrow$ other masks (%)	9.57	**4.42**
Layer 1: [MASK] $\rightarrow$ decoded history (%)	37.05	**48.63**
Layer 3: [MASK] $\rightarrow$ other masks (%)	23.14	**48.54**
Layer 4: [MASK] $\rightarrow$ other masks (%)	14.35	**23.42**
Layer 2: mask-only attention entropy	0.780	**0.931**
Layer 3: mask-only attention entropy	0.696	**0.973**
Layer 4: mask-only attention entropy	0.890	**0.984**

B Additional Mechanistic Analyses

This section provides additional mechanistic analyses of Loop-Distill from three complementary perspectives. Appendix B.1 examines how shallow masked tokens acquire contextual information through attention, Appendix B.2 studies how they are transformed through MoE routing, and Appendix B.3 analyzes whether shallow layers make premature token commitments. Together, these analyses provide a more complete picture of how Loop-Distill improves shallow information flow.

Unless otherwise noted, all analyses in this appendix are conducted on GLM-4.7-Flash with block size 32, consistent with the ablation setup in Section 5.3, and are averaged over GPQA-Diamond, LiveCodeBench-v6, OlympiadBench, and MMLU-Pro.

B.1 Attention-Side Evidence for Shallow Information Flow Refinement

To better understand how Loop-Distill changes shallow masked token refinement, we first examine the attention component of shallow transformer layers for masked positions during decoding. Since attention is the stage where each masked token gathers contextual evidence from decoded history and other positions in the current block, it provides a direct view of how information is acquired before the token is transformed by the subsequent MLP block. We group key tokens into several categories, including other masked positions in the current decoding block, prompt tokens, and already decoded response tokens, and compute the average attention mass assigned to each group.

Table 9 shows that Loop-Distill induces a localized reorganization of shallow attention. In the first layer, masked queries attend less to other unresolved masked positions and more to already decoded response history, indicating that shallow layer refinement is anchored more strongly on reliable observed context. In the following shallow layers, masked-to-masked attention increases substantially, suggesting stronger within-block information exchange once an initial contextual anchor has been established. We also observe higher mask-only attention entropy in these shallow layers, which indicates that attention does not collapse prematurely onto a small set of positions. This is beneficial because shallow masked token representations are still weakly specified and contain little token-specific semantic information. A broader, higher-entropy attention pattern enables each masked position to collect contextual cues from a wider set of unresolved positions, improving early-stage information exchange and reducing the risk of committing to narrow but unreliable interactions.

B.2 MLP-Side Evidence for Shallow Information Flow Refinement

We next examine the MLP component of the same shallow transformer layers. If the attention-side changes above improve how masked tokens gather contextual evidence, a complementary question is whether the subsequent token-wise transformation also becomes more structured. In an MoE backbone, expert routing mediates this transformation. We therefore track, for each masked token, the selected experts at shallow layers over consecutive denoising steps and measure routing stability as the fraction of adjacent denoising-step pairs that preserve the same expert assignment.

Table 10: **MLP-side evidence for shallow information flow refinement.** Loop-Distill increases the temporal consistency of shallow expert assignments for masked positions across denoising steps, especially in the shallow MLP layers. This suggests that partially specified masked tokens are more consistently routed to a specialized subset of experts under information flow refinement.

Layer	Vanilla stability	Loop-Distill stability
1	81.68%	**87.01%**
2	76.97%	**80.79%**
3	67.83%	**72.45%**

Table 11: **Token-level evidence for shallow spurious commitments.** Vanilla exhibits severe shallow token collapse: many active masked positions become spuriously high-confidence on a few unrelated tokens, yet rarely agree with the final prediction. Loop-Distill almost entirely removes these shallow high-confidence collapses. Here, **High-conf.** denotes the fraction of active masked positions whose shallow-layer top-1 confidence exceeds the threshold, and **Agr.** denotes agreement with the final-layer top-1 prediction.

Model	Layer	Conf. threshold	High-conf.	Agr.	Most common token
Vanilla	1	≥ 0.80	56.39%	18.98%	token 220 (single space)
Vanilla	1	≥ 0.95	1.96%	7.45%	token 220 (single space)
Vanilla	2	≥ 0.80	2.42%	1.35%	token 5753 (`#ifdef`)
Vanilla	3	≥ 0.80	1.22%	0.00%	token 26186 (`multi`)
Vanilla	3	≥ 0.95	0.79%	0.00%	token 26186 (`multi`)
Loop-Distill	1	≥ 0.80	0.03%	31.11%	token 220 (single space)
Loop-Distill	1	≥ 0.95	0.00%	–	–
Loop-Distill	2	≥ 0.80	0.00%	–	–
Loop-Distill	3	≥ 0.80	0.00%	–	–

Table 10 shows that Loop-Distill yields more stable routing trajectories, especially in the shallowest MoE layers. That is, for the same masked position, the selected experts in the Loop-Distill model are more consistent from one denoising step to the next than in vanilla.

This finding is complementary to the attention-side evidence in Appendix B.1. Attention shows that Loop-Distill changes how shallow masked tokens collect and exchange contextual information, while routing shows that these partially specified masked states are then more consistently assigned to a specialized subset of experts. Together, the two results support an interpretation: Loop-Distill improves shallow masked token refinement across the full information flow path, from context acquisition in self-attention to expert-specialized transformation in the MLP block.

B.3 Token-Level Evidence for Suppressing Shallow Spurious Commitments

The main text shows that shallow layers in vanilla converted DLM often make overconfident predictions that disagree with the final output. To probe this behavior more directly, we inspect token identities at shallow layers using the shared LM head. Specifically, for active masked positions during decoding, we record shallow-layer top-1 predictions whose confidence exceeds a fixed threshold and compare them with the final-layer top-1 predictions.

Table 11 reveals that the shallow overconfidence problem in Vanilla is not merely a calibration issue, but a token-level collapse to spurious high-confidence outputs. At layer 1, a large fraction of high-confidence masked positions collapse to the same whitespace token, while later shallow layers often over-predict a small set of unrelated subword fragments. Despite their high confidence, these token choices rarely agree with the final-layer prediction.

Loop-Distill sharply reduces this behavior. The ratio of shallow positions with high confidence drops dramatically, and the concentration on a few spurious token types nearly disappears. This result strengthens the interpretation in the main text: Loop-Distill does not simply lower confidence uniformly, but specifically suppresses premature token commitments that arise before sufficient contextual evidence has been integrated.

Table 12: **Complementarity with AR-representation alignment.** Results on Qwen-30B-A3B-2507 with block size 4. Method abbreviations are: **Base** = baseline, **ARL** = AR loss, **LD** = Loop-Distill, and **LD+ARL** = Loop-Distill + AR loss, Δ = accuracy gains over base, Avg. = average. Benchmark abbreviations are: **IFB** = IFBench, **KB** = KorBench, **GPQ** = GPQA-Diamond, **BBH** = BBH, **MMP** = MMLU-Pro, **MUS** = MuSR, **MBP** = MBPP+, **HE** = HumanEval+, **BCB** = BigCodeBench, **LCB** = LiveCodeBench-v6, **MAT** = MATH, **OLY** = OlympiadBench, and **ZEB** = ZebraLogic. Loop-Distill remains beneficial over AR loss, indicating that the two methods are complementary.

Method	Avg.	IFB	KB	GPQ	BBH	MMP	MUS	MBP	HE	BCB	LCB	MAT	OLY	ZEB
Base	56.53	31.07	55.16	44.41	74.20	57.92	72.16	75.93	79.57	34.47	24.39	87.40	54.26	43.97
ARL	57.93	32.82	57.14	47.87	75.29	58.87	73.21	77.06	81.41	33.96	25.84	88.07	**56.32**	45.18
Δ	*+1.40*	*+1.75*	*+1.98*	*+3.46*	*+1.09*	*+0.95*	*+1.05*	*+1.13*	*+1.84*	*-0.51*	*+1.45*	*+0.67*	*+2.06*	*+1.21*
LD	58.62	32.40	56.64	45.04	75.34	60.12	72.84	76.85	**83.38**	**38.68**	26.87	88.62	55.52	**49.78**
Δ	*+2.09*	*+1.33*	*+1.48*	*+0.63*	*+1.14*	*+2.20*	*+0.68*	*+0.92*	*+3.81*	*+4.21*	*+2.48*	*+1.22*	*+1.26*	*+5.81*
LD+ARL	**59.87**	**35.30**	**59.04**	**48.93**	**77.21**	**62.14**	**74.80**	**79.88**	83.23	36.58	**29.41**	**90.26**	55.48	46.05
Δ	*+3.34*	*+4.23*	*+3.88*	*+4.52*	*+3.01*	*+4.22*	*+2.64*	*+3.95*	*+3.66*	*+2.11*	*+5.02*	*+2.86*	*+1.22*	*+2.08*

C Relation to AR Representation Alignment

Our method is related to prior efforts on converting autoregressive (AR) language models into diffusion language models (DLMs), but it targets a different part of the conversion gap. A representative direction introduces an auxiliary alignment loss to preserve the hidden-state geometry of the source AR model during conversion (Fu et al., 2026). Such methods are intended to reduce the representation shift induced by moving from next-token prediction to masked parallel denoising.

Loop-Distill focuses on a different mechanism. Rather than aligning the converted DLM to the original AR hidden states, it improves the internal refinement dynamics of masked token representations within the DLM itself, especially at shallow layers where we observe premature stabilization and overconfident token preferences. Conceptually, these two strategies act at different levels: AR alignment regularizes cross-paradigm representation drift, whereas Loop-Distill improves within-model masked state information flow. This suggests that they may be complementary.

We test this hypothesis using one representative AR-alignment objective. As shown in Table 12, on Qwen-30B-A3B-2507 with block size 4, we add an AR alignment loss (*AR loss*) with weight 0.3 following the conversion recipe of Fu et al. (2026). Starting from the same baseline, AR loss alone improves the 13-benchmark average from 56.53 to 57.93 (+1.39), while Loop-Distill alone improves it to 58.62 (+2.09). Combining the two further raises the average to 59.87, corresponding to a gain of +3.34 over the baseline and +1.96 over AR loss alone.

These results support two takeaways. First, Loop-Distill remains effective when combined with this representative AR-to-DLM alignment objective, demonstrating robustness beyond the vanilla conversion setting. Second, the additional gain from the combination suggests that Loop-Distill captures improvements that are not redundant with this AR-alignment signal.

D Layer-wise Quantitative Comparison of Masked State Changes

To complement the summary statistics in Table 7, we provide a fuller layer-wise view of how masked token representations evolve across depth. For each adjacent layer pair $(\ell, \ell+1)$, we compute the average cosine similarity

$$S_\ell = \mathbb{E}_i\left[\cos(h_i^{(\ell)}, h_i^{(\ell+1)})\right], \tag{19}$$

where the expectation is taken over masked positions that are eventually committed during diffusion decoding. A lower value of S_ℓ indicates a larger representation change between the two layers, while a higher value indicates that the masked states have already become relatively stable.

Figure 4 compares the full layer-wise cosine-similarity curves of vanilla AR-to-BD and Loop-Distill. The top panels show that the vanilla exhibits an extremely sharp first transition, after which masked states become rapidly over-stable through the remaining shallow layers. By contrast, Loop-Distill

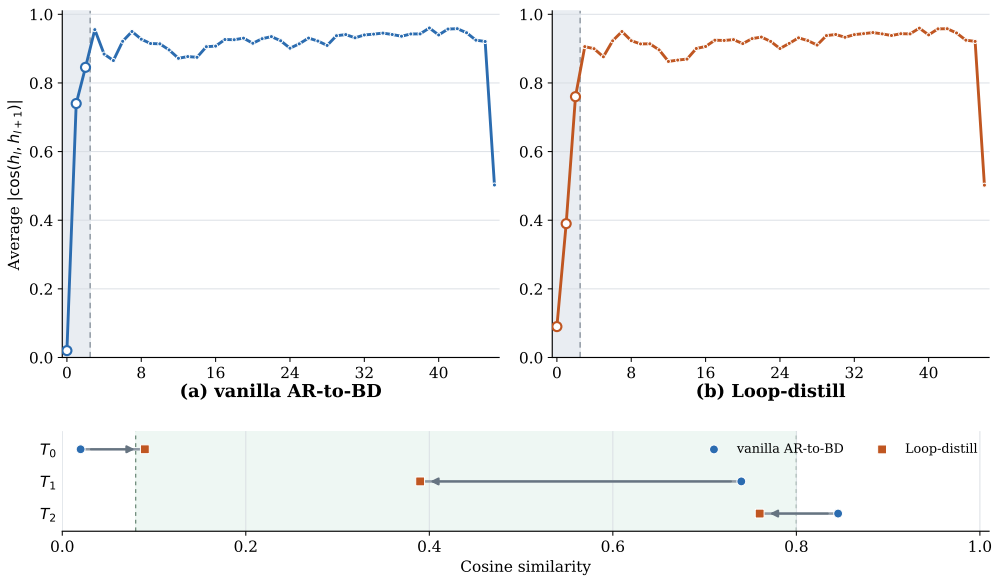


Figure 4: **Layer-wise quantitative comparison of masked token state changes.** Top: average cosine similarity between adjacent-layer masked token states, $\cos(h_i^{(\ell)}, h_i^{(\ell+1)})$, across model depth for vanilla AR-to-BD and Loop-Distill. Lower cosine similarity indicates a larger inter-layer update. Vanilla shows an extremely sharp first transition followed by premature stabilization, whereas Loop-Distill makes the first update less abrupt and preserves stronger refinement in the following shallow transitions. Bottom: zoomed comparison of the first three transitions (T_0, T_1, T_2), highlighting the redistribution of shallow refinement under Loop-Distill.

reduces the abruptness of the first transition and redistributes a larger fraction of refinement into the subsequent shallow transitions. The bottom panel highlights the shallow layers' transitions directly, making the redistribution effect more explicit: compared with the vanilla, Loop-Distill yields a larger S_0 but substantially smaller S_1, indicating a milder first update followed by stronger refinement.

This layer-wise quantitative picture is consistent with the main claim of the paper. Loop-Distill does not simply increase or decrease representation change uniformly across depth. Instead, it reshapes the *timing* of shallow masked token refinement: the first update becomes less disruptive, while the next shallow transition carries more meaningful contextual refinement.

E More Implementation Details

Evaluation protocol. Unless otherwise noted, all reported accuracy numbers are averaged over three repeated evaluation runs under the same decoding configuration (threshold: $\tau = 0.95$).

Training setup. For the main comparison between Vanilla and Loop-Distill, we keep the supervised fine-tuning (SFT) setup matched across methods. Both models use the same optimizer and distributed setup. Vanilla requires approximately 7.5 hours of training, whereas Loop-Distill requires approximately 8.2 hours, corresponding to an additional training wall-clock overhead of roughly 10%. Importantly, this overhead is incurred only during the SFT. Loop-Distill does not modify the deployed model architecture and introduces no extra inference-time loop. This one-time training overhead translates into an average throughput improvement of 11% at inference, together with consistent accuracy gains. We therefore consider the added training cost to be practically modest.

Loop configuration pre-computing. The loop configuration is determined *before training*, rather than tuned online during Loop-Distill. Specifically, we run the initial AR checkpoint under DLM mode and use the local masked state refinement diagnostics in Section 4.3 to choose both the loop placement and the loop depth. These choices are then kept fixed throughout training.