TOWARDS UNBIASED ON-POLICY DISTILLATION FOR BLOCK DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) has emerged as an effective post-training paradigm for language models, with recent efforts extending it to block diffusion language models (BDLMs). However, existing studies focus almost exclusively on small block sizes, leaving distillation into student models with larger blocks underexplored. In this work, we investigate this regime and reveal two critical optimization biases that induce severe training instability. First, mismatched block boundaries between teacher and student cause ***context misalignment***, providing distorted supervisory signals that misguide student decoding. Second, even under aligned contexts, an ***intrinsic optimization bias*** in OPD, where the student tends to rapidly absorb high-support signals while lagging on low-support updates, drives a premature confidence surge that traps weaker students in catastrophic overconfidence collapse. To resolve these problems, we propose **Un-OPD**, an unbiased on-policy distillation framework with two novelties for stabilizing BDLM training. First, Un-OPD introduces a boundary-aware step filtering strategy that eliminates context-misaligned decoding steps. Second, Un-OPD proposes moderating optimization intensity at high-support positions via a support-rebalanced confidence calibration, thereby bypassing overconfidence collapse. Beyond stability, we also introduce a rollout reuse mechanism to reduce rollout generation overhead. Extensive experiments on math reasoning and code generation benchmarks show that Un-OPD consistently stabilizes training and delivers superior performance while reducing wall-clock training time by approximately half.

1 INTRODUCTION

Diffusion large language models (dLLMs) have recently emerged as a compelling alternative to autoregressive (AR) models (Achiam et al., 2023; Hui et al., 2024; Yang et al., 2025) by generating multiple tokens in parallel through iterative denoising (Nie et al., 2025; Ye et al., 2025). For example, some open-source models like LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) maintain a long sequence of mask tokens and iteratively reveal tokens through parallel denoising steps. To inherit the advantages of AR models, particularly KV caching compatibility (Kwon et al., 2023; Ni et al., 2026) and flexible sequence extension, recent dLLM works have advanced toward block diffusion large language models (BDLMs) (Arriola et al., 2025; Cheng et al., 2026; Tian et al., 2025; Wu et al., 2026b; Bie et al., 2025; 2026). By decomposing a sequence into consecutive blocks, BDLMs perform global reasoning autoregressively across blocks while generating tokens via parallel diffusion within each block, achieving a superior quality–latency trade-off.

Following the success of post-training in AR models, particularly reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; Yu et al., 2026), reinforcement learning (RL) has been actively adapted to enhance the reasoning capabilities of dLLMs (Pan et al., 2026a; Zhao et al., 2026; Wang et al., 2026b; Ou et al., 2026; Ma et al., 2026; Ni et al., 2026). Specifically, to bypass the intractable conditional likelihood estimation inherent in diffusion denoising, pioneering efforts introduce mean-field surrogates (Zhao et al., 2026; Tang et al., 2026) or tree-structured policy updates (Pan et al., 2026a). Nevertheless, these approaches either rely on coarse outcome-based rewards leading to inaccurate and sparse credit assignment across multi-step denoising trajectories (Shao et al., 2026; Zhao et al., 2026), or incur substantial inference overhead to acquire finer-grained reward signals (Pan et al., 2026a). Recently, on-policy distillation (OPD) (Hou et al., 2026; Yang et al., 2026b) has emerged as another promising post-training paradigm. By aligning the stu-

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.

dent with teacher predictions along its own rollout trajectories, OPD provides dense, token-level supervision. Recent studies (Hou et al., 2026; Yang et al., 2026b; Fu et al., 2026) demonstrate that OPD outperforms previous RLVR algorithms on language models in terms of both optimization stability and sample efficiency.

Despite extensive progress in AR models, OPD for dLLMs remains at an early stage. dOPSD (Dat et al., 2026) extends self-distillation to full-sequence denoising models such as LLaDA (Nie et al., 2025) using predictions from later states with more revealed tokens to supervise earlier states. This design is less applicable to BDLMs, where later states reveal context only within the current block. TOPD (Ren et al., 2026) extends OPD to BDLMs by distilling along the student's denoising trajectories, but it focuses exclusively on small block sizes, leaving distillation into students with larger blocks, which offer inherently greater parallel decoding potential, largely underexplored. In this work, we investigate scaling OPD to BDLMs with larger block sizes. We use small-block BDLMs as teachers because they provide higher-quality generations, and distill them into students with varying block sizes.

Our work uncovers two underlying optimization biases that impede the effective distillation of BDLMs: **(1) Context misalignment.** When denoising within a larger block, the student receives biased supervision because the teacher cannot access its authentic decoding states across mismatched boundaries. Such signals tend to be overly aggressive or excessively conservative, disrupting the student's confidence-driven decisions. **(2) Biased learning dynamics.** During OPD, the student prioritizes tokens with strong teacher support over weakly supported ones, causing confidence to prematurely surge and then fall back. This spurious surge drives fragile students into low-quality decoding regimes that are difficult to rectify. To address these two problems, we propose **Un-OPD** (Unbiased On-Policy Distillation), a simple framework for scaling OPD to BDLMs with larger block sizes. Specifically, we first restrict OPD to decoding states with context alignment, ensuring impartial teacher supervision signals. We then quantify *teacher support* to distinguish strongly supported positions and rebalance the teacher support by moderating the update intensity, preventing overconfidence collapse. Beyond stability, we further introduce a *rollout reuse* mechanism that reuses sampled trajectories for reducing rollout generation overhead.

In summary, the main contributions of this work are as follows:

- We pioneer the study of OPD on BDLMs with varying block sizes, identifying two underlying optimization bottlenecks: *context misalignment* arising from mismatched block sizes and *biased learning dynamics* inherent to standard OPD.
- We propose **Un-OPD**, an unbiased OPD framework that integrates context-aligned step selection and support-rebalanced confidence calibration to stabilize the training dynamics. We also introduce a rollout reuse mechanism to reduce rollout overhead during training.
- We conduct a comprehensive evaluation across various block sizes and demonstrate that Un-OPD achieves superior performance over counterparts, alongside greater training efficiency.

2 Related Work

Block Diffusion Language Models. dLLMs (Nie et al., 2025; Zhu et al., 2025; Ye et al., 2025) offer an alternative to autoregressive LLMs (Yang et al., 2025; Achiam et al., 2023) through iterative discrete denoising, with recent gains from scaling data and model size. LLaDA (Nie et al., 2025; Zhu et al., 2025) trains from scratch via masking and unmasking, whereas Dream (Ye et al., 2025) adapts pretrained autoregressive models. Full-sequence dLLMs, however, struggle to combine coherent chain-of-thought reasoning with efficient inference mechanisms such as KV caching (Kwon et al., 2023; Ni et al., 2026). Block diffusion language models (BDLMs) (Arriola et al., 2025; Cheng et al., 2026; Tian et al., 2025; Wu et al., 2026b; Bie et al., 2025; 2026) address this trade-off by generating blocks autoregressively and denoising tokens in parallel within each block. Typically adapted from autoregressive models, BDLMs aim to retain their reasoning capabilities while enabling parallel intra-block generation.

On-Policy Distillation. By aligning student-generated rollouts with per-token teacher predictions, on-policy distillation (OPD) (Hou et al., 2026; Yang et al., 2026b; Gu et al., 2026; Fu et al., 2026; Xie et al., 2026; Xu et al., 2026; Zhang et al., 2026; Jin et al., 2026; Wang et al., 2026a; Xing

et al., 2026) has demonstrated superior post-training efficiency compared to GRPO-style optimization (Shao et al., 2024; Yu et al., 2026; Gao et al., 2025; Tan et al., 2025) and off-policy counterparts (Wang et al., 2023; Ren & Sutherland, 2025; Wu et al., 2026a). To extract more reliable teacher supervision, several works (Zhang et al., 2026; Wang et al., 2026a; Xu et al., 2026; Xie et al., 2026; Zheng et al., 2026) focus optimization on student rollout prefixes. In parallel, on-policy self-distillation (OPSD) (Lu et al., 2026; Yang et al., 2026a; Pan et al., 2026b; Yuan et al., 2026) uses the same model as the teacher, conditioned on privileged context such as ground-truth answers. TOPD (Ren et al., 2026) extends OPD to BDLMs (Cheng et al., 2026) with small block sizes. We investigate instability at larger student block sizes and address context misalignment and biased optimization dynamics.

Efficient RLVR via Rollout Replay. To improve data efficiency in RLVR, recent works incorporate rollout replay into GRPO-style optimization, curbing policy drift via on/off-policy interleaving (Zhang et al., 2025; Zhan et al., 2026; Li et al., 2025), quality filtering (Zhang et al., 2025; Wang et al., 2025), or importance-sampling corrections (Mao et al., 2026). However, these methods require delicate heuristics to manage staleness and off-policy bias. In contrast, we extend rollout replay to OPD, which exhibits fundamentally higher tolerance to trajectory staleness. This allows Un-OPD to employ an aggressive replay schedule that substantially reduces rollout generation overhead without the complexity of trace filtering or data mixing.

3 METHOD

We propose **Un-OPD**, a framework that scales OPD to BDLMs with large block sizes by integrating two recipes: context-aligned distillation and support-rebalanced distillation. Our design is driven by two key sources of training instability: *context misalignment* arising from mismatched block boundaries and *biased learning dynamics* inherent to standard OPD. In this section, we first provide the preliminaries in Sec. 3.1. Next, Sec. 3.2 analyzes context misalignment and introduces context-aligned distillation. Then, Sec. 3.3 investigates the biased learning dynamics of OPD and introduces our support rebalance strategy. Finally, Sec. 3.4 unifies these components into the overall Un-OPD training framework while introducing rollout reuse to reduce trajectory-sampling overhead.

3.1 PRELIMINARIES

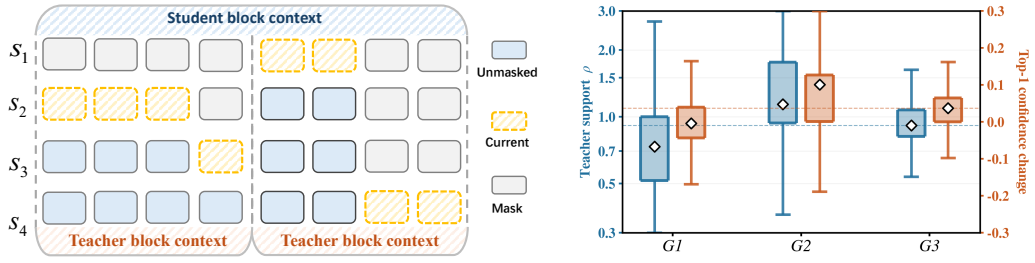
On-policy distillation. On-policy distillation (OPD) trains a student policy π_θ on its own rollouts using token-level supervision from a fixed teacher policy π_{T}. Given a prompt $q \sim \mathcal{D}$, the student generates a response $\tau = (o_1, \ldots, o_{|\tau|})$ where $|\tau|$ is the length of the trajectory. Let c_i denote the conditioning context at which token o_i is predicted, including the prompt and the visible decoding state. In the reverse-KL formulation considered here, the student minimizes:

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|q)} \left[\frac{1}{|\tau|} \sum_{i=1}^{|\tau|} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid c_i) \,\|\, \pi_{\mathrm{T}}(\cdot \mid c_i)) \right], \tag{1}$$

where sampled rollout contexts are held fixed during each optimization update. The reverse KL at position i is computed over the vocabulary $\mathcal{V}$:

$$\mathcal{L}_i = \sum_{v\in\mathcal{V}} \pi_\theta(v \mid c_i) \big[\log \pi_\theta(v \mid c_i) - \log \pi_{\mathrm{T}}(v \mid c_i) \big]. \tag{2}$$

Trace-aligned OPD for block diffusion language models (BDLMs). BDLMs generate sequences in a semi-autoregressive fashion: they proceed block by block, iteratively revealing tokens within the current block while conditioning on all preceding, fully generated blocks. Following TOPD (Ren et al., 2026) and TraceRL (Wang et al., 2026b) (a GRPO-based method for block diffusion language models), we tailor OPD to this decoding paradigm by supervising the student directly along its on-policy denoising trajectory. Specifically, given a prompt q, the student samples an execution trace $\xi = (s_0, a_0, s_1, \ldots, a_{\mathrm{K}-1}, s_{\mathrm{K}})$, where s_k denotes the intermediate decoding state (with partially unmasked tokens), $a_k = \{(i, x_i)\}$ represents the set of token decisions committed at step k, and s_{K} is the finalized sequence. For each committed position $(i, x_i) \in a_k$, we evaluate $\pi_{\mathrm{T}}(\cdot \mid q, s_k, i)$ and $\pi_\theta(\cdot \mid q, s_k, i)$ and minimize their reverse-KL divergence.



(a) (b)

Figure 1: **Context misalignment and its effects on OPD. (a)** The student denoises an 8-token block, while the teacher provides supervision within 4-token blocks. Contexts are misaligned at states s_1–s_3 and aligned at s_4. **(b)** Box plots show distributions of teacher support ρ (blue) and post-OPD top-1 probability change (orange) for context regimes G_1, G_2, and G_3. The diamonds mark mean values.

3.2 Context Aligned Distillation

Understanding context misalignment. Unlike autoregressive teacher–student pairs with identical causal contexts, BDLMs with different block sizes can have misaligned visible contexts even under trace-aligned OPD. Fig. 1(a) illustrates this mismatch for teacher and student block sizes of 4 and 8. The student undergoes four commit steps within its 8-token block (yielding states s_1–s_4), whereas the teacher's span is split into two 4-token blocks by the red dashed boundary. Consequently, their contexts align only at s_4, while s_1–s_3 suffer from distinct mismatches: at s_1, the teacher conditions on the fully generated left-block context that remains noisy to the student, whereas at s_2 and s_3, the student accesses unmasked context from the right-block that lies strictly beyond the teacher's first block boundary.

Optimization bias under context misalignment. To probe how context misalignment affects student optimization, we pair an `SDAR-4B-b8` student with a `TraDo-8B-b4` teacher and categorize decoding states into three regimes: G_1 (student sees more context such as s_2, s_3), G_2 (teacher sees more such as s_1), and G_3 (contexts are aligned such as s_4). Let $h_i = \arg\max_v p_{\mathrm{S},i}(v)$ denote the student's top-1 token at position i. We quantify the teacher support $\rho_i = p_{\mathrm{T},i}(h_i)/p_{\mathrm{S},i}(h_i)$, where $\rho_i > 1$ and $\rho_i < 1$ denote stronger and weaker teacher support for the student's top-1 prediction, respectively. In Fig. 1(b), the blue box plots show the teacher support distributions across the three state groups, while the orange box plots show the distributions of changes in the student's top-1 probability after training with OPD exclusively on each group. The teacher support skews predominantly below 1.0 in G_1 and above 1.0 in G_2, whereas G_3 remains well-balanced. The confidence change mirrors this pattern: G_1 yields negligible confidence change, G_2 causes a sharp surge, and G_3 produces a moderate rise. These results demonstrate that context asymmetry systematically biases teacher supervision signals on the student rollout tokens in two opposite directions:

- **Under-supportive supervision signals.** With less visible context than the student, the teacher tends to underestimate the student's valid rollouts relative to aligned contexts, which leaves the student at risk of **failing to reinforce confidence in correct predictions** through OPD.
- **Over-supportive supervision signals.** With more visible context than the student, the teacher tends to provide overly aggressive support relative to aligned contexts, which puts the student at risk of **becoming overconfident in incorrect predictions** through OPD.

Context aligned step selection. Let $s_{\mathrm{T},k}$ and $s_{\mathrm{S},k}$ denote the teacher and student block states at step k. To mitigate context-induced bias, we retain only decisions committed at steps with aligned visible contexts:

$$\mathcal{A}_{\text{align}} = \big\{(k,i) : (i,x_i) \in a_k,\ 0 \le k < K,\ s_{\mathrm{T},k} = s_{\mathrm{S},k}\big\}. \tag{3}$$

Here, (k,i) denotes position i committed at step k. Averaging the reverse KL over these positions gives the context-aligned OPD objective:

$$\mathcal{L}_{\text{aligned-OPD}} = \frac{1}{|\mathcal{A}_{\text{align}}|} \sum_{(k,i)\in\mathcal{A}_{\text{align}}} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid q, s_{\mathrm{S},k}, i) \,\|\, \pi_{\mathrm{T}}(\cdot \mid q, s_{\mathrm{T},k}, i)). \tag{4}$$

3.3 SUPPORT-REBALANCED DISTILLATION

Though we have identified and mitigated context-induced bias in Sec. 3.2, training remains suboptimal. Crucially, we observe that another bias, the internal optimization dynamics of OPD, also compromises BDLM training stability. Here, we first present empirical observations of this behavior in OPD, then provide a theoretical grounding, and finally introduce a simple yet effective solution.

Biased optimization dynamics in OPD. Since top-1 token confidence critically dictates decoding decisions in BDLMs, we begin by examining the evolution of the student's average top-1 confidence throughout the entire OPD training process. As illustrated in Fig. 2, decoding confidence of the student model exhibits a distinct two-stage pattern throughout training: *a rapid surge during the early phase, followed by a slight decline during later training.* To dissect the underlying mechanism behind this phenomenon, we leverage the previously defined teacher support ρ_i in Sec. 3.2 given its tight correlation with confidence shifts, and divide the supervised positions into two distinct groups: $\mathcal{I}_+ = \{i : \rho_i > 1\}$ and $\mathcal{I}_- = \{i : \rho_i < 1\}$. Empirically, updates from $\mathcal{I}_+$ tend to increase the student's decoding confidence, whereas updates from $\mathcal{I}_-$ tend to decrease it.

As shown in Fig. 2, we reveal that the fraction of high-support positions steadily decreases, while that of low-support positions rises before eventually plateauing. We hypothesize that this early confidence surge stems from an asymmetric learning dynamic: the student **rapidly internalizes** supervisory signals from high-support positions ($\mathcal{I}_+$), whereas updates from low-support positions ($\mathcal{I}_-$) **suffer from an optimization lag**. We term this learning asymmetry the *intrinsic optimization bias* of OPD. Consequently, as the student's confidence is prematurely driven up, the rollout policy becomes increasingly prone to sampling tokens beyond the teacher's support region, which in turn produces more low-support positions that ultimately dominate training and trigger a subsequent pullback in confidence.

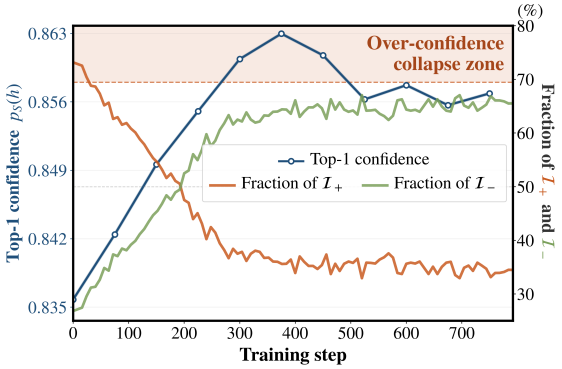


Figure 2: Evolution of student mean top-1 confidence (blue) alongside subset fractions of $\mathcal{I}_+$ (orange) and $\mathcal{I}_-$ (green). The shaded area denotes the overconfidence collapse zone.

Theoretical grounding. To theoretically characterize this optimization bias, we examine the reverse-KL gradient update to the logit of the student's top-1 prediction h_i. Let $a_i = p_{\mathrm{S},i}(h_i)$ and $b_i = p_{\mathrm{T},i}(h_i)$, such that $\rho_i = b_i/a_i$. We denote by $D_i^{\mathrm{tail}} \geq 0$ the reverse-KL divergence between the renormalized tail distributions of the student and teacher over all non-top-1 tokens ($v \neq h_i$).

Proposition 1 (Asymmetric top-1 logit updates under reverse-KL) *For a fixed context and a teacher distribution with positive token probabilities, a reverse-KL gradient descent step with learning rate $\eta > 0$ directly on the student's softmax logits yields:*

$$\Delta z_{\mathrm{S},i}(h_i) = \eta a_i(1-a_i)\left[\log \frac{b_i(1-a_i)}{a_i(1-b_i)} + D_i^{\mathrm{tail}}\right]. \tag{5}$$

The proof is provided in Appendix D.1.

Proposition 1 shows that the nonnegative tail term D_i^{tail} reinforces upward logit updates for $\rho_i > 1$, but can weaken or reverse downward updates for $\rho_i < 1$. This local logit-level analysis provides a mechanism consistent with the observed asymmetry, although actual training dynamics also depend on the overall training data distribution.

Support-rebalanced confidence calibration. Due to this optimization bias, OPD is not a free lunch for BDLMs. When weaker students are guided by a strong teacher, a premature surge in confidence readily pushes them into a degenerate generation regime (Fig. 2). Consequently, such students slip into a catastrophic overconfidence zone, failing to recover even in later training stages to a confidence regime commensurate with their actual capacity. Here, we propose calibrating the supervision signals specifically at high-support positions. In implementation, we temper the teacher target by interpolating its logits with the student's detached logits for each $i \in \mathcal{I}_+$, thereby moderating update

intensity at high-support positions. Let $\mathcal{I}$ denote the set of supervised token positions and $\lambda \in [0, 1]$ be the teacher interpolation weight at high-support positions. The calibrated teacher target and the resulting reverse-KL objective are defined as:

$$\widetilde{z}_{\mathrm{T},i}(v) = \left[1 - (1-\lambda)\mathbf{1}_{\{i \in \mathcal{I}_+\}}\right] z_{\mathrm{T},i}(v) + (1-\lambda)\mathbf{1}_{\{i \in \mathcal{I}_+\}}\, \mathrm{sg}[z_{\mathrm{S},i}(v)]\,,$$
$$\mathcal{L}_\lambda = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} D_{\mathrm{KL}}(p_{\mathrm{S},i} \,\|\, \mathrm{softmax}(\widetilde{z}_{\mathrm{T},i}))\,. \tag{6}$$

Here, $\mathbf{1}_{\{i \in \mathcal{I}_+\}}$ indicates a high-support position and $\mathrm{sg}[\cdot]$ denotes stop-gradient. At these positions, the teacher and detached student logits receive weights λ and $1 - \lambda$, respectively; smaller λ yields stronger calibration. Low-support positions retain the original teacher target.

3.4 Overall Framework

Overall objective. Unifying the two training recipes, we can obtain the unbiased OPD framework for BDLMs, whose overall objective minimizes the reverse-KL divergence over aligned decisions in $\mathcal{A}_{\mathrm{align}}$ using the calibrated teacher target:

$$\mathcal{L}_{\text{Un-OPD}}(\theta) = \mathbb{E}_{q,\xi}\left[\frac{1}{|\mathcal{A}_{\mathrm{align}}|} \sum_{(k,i) \in \mathcal{A}_{\mathrm{align}}} D_{\mathrm{KL}}(p_{\mathrm{S},k,i} \,\|\, \mathrm{softmax}(\widetilde{z}_{\mathrm{T},k,i}))\right], \tag{7}$$

where $p_{\mathrm{S},k,i} = \pi_\theta(\cdot \mid q, s_{\mathrm{S},k}, i)$ and $\widetilde{z}_{\mathrm{T},k,i}$ is the calibrated target from Eq. 6. When $\lambda = 1$, Eq. 7 recovers context-aligned OPD (Eq. 4), while $\lambda < 1$ calibrates the teacher target at high-support positions to curb premature confidence inflation.

Rollout reuse. We notice that OPD is inherently better suited for trajectory reuse compared to RLVR. In OPD, historical rollouts serve merely as conditioning states rather than frozen discrete actions or stale scalar rewards as in RLVR. Across multiple optimization steps on the same rollout batch, we recompute the student distributions under the current policy, while the frozen teacher distributions over the fixed states remain unchanged. The calibrated targets and distillation gradients are recomputed accordingly. Consequently, the primary discrepancy in OPD replay arises solely from *state distribution mismatch* between the sampling policy and the updated student. Unlike RLVR, which suffers from volatile importance sampling and reward clipping over historical rollouts, OPD suffers from substantially milder off-policy degradation. This insight motivates our *rollout reuse* mechanism, which performs multiple updates per rollout to effectively reduce the heavy cost of rollout sampling. The complete Un-OPD training procedure is presented in Alg. 1 of appendix A.1.

4 Experiments

In this section, we conduct comprehensive experiments across BDLMs with varying block sizes to evaluate the effectiveness of Un-OPD. We first detail the experimental configurations in Sec. 4.1, then assess how the proposed Un-OPD improves performance across different benchmarks in Sec. 4.2. Finally, we provide an ablation study about the core recipes of Un-OPD in Sec. 4.3.

4.1 Experimental Settings

Baselines and Benchmarks. To comprehensively evaluate the scalability and robustness of Un-OPD, we employ `SDAR-4B` and `SDAR-1.7B` (Cheng et al., 2026) with varying block sizes ($b \in \{4, 8\}$) as student BDLMs, paired with a default `TraDo-8B` (Wang et al., 2026b) as teacher ($b = 4$). We evaluate the models on standard benchmarks for mathematical reasoning (*e.g.,* GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC 2023), as well as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation. To ensure fair comparisons, we evaluate against **SFT**, **TraceRL** (Wang et al., 2026b), and **TOPD** (Ren et al., 2026) as three primary baselines.

Implementation details. For mathematical reasoning, we follow TraceRL (Wang et al., 2026b) and use 8K math problems for post-training. For code generation, we use a curated subset of over $10,000$ problems from AceCoder (Zeng et al., 2025). Student trajectories are collected using dynamic decoding with a batch size of 64 prompts (one rollout per prompt). We adopt the dynamic decoding

Table 1: Performance comparison on math and code generation benchmarks. We report accuracy (%) with gains over the base model computed by direct subtraction. With the teacher block size fixed at 4, student block sizes of 8 and 4 correspond to **asymmetric** and **symmetric** configurations, respectively. The best results in each group are **boldfaced** and underlined.

Method	Mathematical Reasoning					Code Generation		
	GSM8K	MATH500	AMC2023	Olympiad	Avg.	MBPP	HumanEval	Avg.
Student Model: SDAR-4B-b8								
Base	88.9	66.3	42.5	29.2	56.7	53.5	67.9	60.7
+ SFT	$88.1_{-0.8}$	$65.4_{-0.9}$	$40.3_{-2.2}$	$27.9_{-1.3}$	$55.4_{-1.3}$	$51.7_{-1.8}$	$65.5_{-2.4}$	$58.6_{-2.1}$
+ TraceRL	$89.5_{+0.6}$	$69.4_{+3.1}$	**$46.8_{+4.3}$**	$30.0_{+0.8}$	$58.9_{+2.2}$	$54.1_{+0.6}$	$68.9_{+1.0}$	$61.5_{+0.8}$
+ TOPD	$88.4_{-0.5}$	$67.1_{+0.8}$	$43.6_{+1.1}$	$30.8_{+1.6}$	$57.5_{+0.8}$	$52.3_{-1.2}$	$65.1_{-2.8}$	$58.7_{-2.0}$
+ Un-OPD (Ours)	**$90.0_{+1.1}$**	**$70.1_{+3.8}$**	$45.8_{+3.3}$	**$32.8_{+3.6}$**	**$59.7_{+3.0}$**	**$55.6_{+2.1}$**	**$71.1_{+3.2}$**	**$63.4_{+2.7}$**
Student Model: SDAR-1.7B-b8								
Base	74.8	51.9	25.9	17.3	42.5	41.2	52.4	46.8
+ SFT	$73.4_{-1.4}$	$50.1_{-1.8}$	$23.3_{-2.6}$	$15.7_{-1.6}$	$40.6_{-1.9}$	$40.5_{-0.7}$	$53.5_{+1.1}$	$47.0_{+0.2}$
+ TraceRL	**$75.6_{+0.8}$**	$53.0_{+1.1}$	**$27.8_{+1.9}$**	$18.3_{+1.0}$	$43.7_{+1.2}$	**$43.5_{+2.3}$**	$54.4_{+2.0}$	**$49.0_{+2.2}$**
+ TOPD	$74.4_{-0.4}$	$50.9_{-1.0}$	$25.5_{-0.4}$	$17.2_{-0.1}$	$42.0_{-0.5}$	$40.1_{-1.1}$	$50.9_{-1.5}$	$45.5_{-1.3}$
+ Un-OPD (Ours)	$75.3_{+0.5}$	**$54.9_{+3.0}$**	$27.7_{+1.8}$	**$18.6_{+1.3}$**	**$44.1_{+1.6}$**	$42.4_{+1.2}$	**$55.6_{+3.2}$**	**$49.0_{+2.2}$**
Student Model: SDAR-4B-b4								
Base	88.1	66.5	43.1	29.5	56.8	50.8	65.2	58.0
+ SFT	**$89.8_{+1.7}$**	$67.4_{+0.9}$	$40.2_{-2.9}$	$31.7_{+2.2}$	$57.3_{+0.5}$	$54.8_{+4.0}$	**$71.3_{+6.1}$**	$63.1_{+5.1}$
+ TraceRL	$87.1_{-1.0}$	$70.4_{+3.9}$	$44.2_{+1.1}$	$30.7_{+1.2}$	$58.1_{+1.3}$	$56.4_{+5.6}$	$71.2_{+6.0}$	$63.8_{+5.8}$
+ TOPD	$89.7_{+1.6}$	$70.8_{+4.3}$	$45.0_{+1.9}$	$32.9_{+3.4}$	$59.6_{+2.8}$	$56.8_{+6.0}$	$70.8_{+5.6}$	$63.8_{+5.8}$
+ Un-OPD (Ours)	$89.5_{+1.4}$	**$72.0_{+5.5}$**	**$45.4_{+2.3}$**	**$33.5_{+4.0}$**	**$60.1_{+3.3}$**	**$57.1_{+6.3}$**	$71.0_{+5.8}$	**$64.1_{+6.1}$**
Student Model: SDAR-1.7B-b4								
Base	74.4	53.2	26.7	16.8	42.8	41.2	53.0	47.1
+ SFT	$76.4_{+2.0}$	$54.2_{+1.0}$	$28.8_{+2.1}$	$19.5_{+2.7}$	$44.7_{+1.9}$	$43.9_{+2.7}$	$56.1_{+3.1}$	$50.0_{+2.9}$
+ TraceRL	**$76.7_{+2.3}$**	$55.9_{+2.7}$	$30.6_{+3.9}$	$19.0_{+2.2}$	$45.6_{+2.8}$	$45.4_{+4.2}$	$55.9_{+2.9}$	$50.7_{+3.6}$
+ TOPD	$76.6_{+2.2}$	$57.7_{+4.5}$	$30.6_{+3.9}$	$21.1_{+4.3}$	$46.5_{+3.7}$	$43.8_{+2.6}$	$57.4_{+4.4}$	$50.6_{+3.5}$
+ Un-OPD (Ours)	$76.1_{+1.7}$	**$58.8_{+5.6}$**	**$31.5_{+4.8}$**	**$21.6_{+4.8}$**	**$47.0_{+4.2}$**	**$46.1_{+4.9}$**	**$57.9_{+4.9}$**	**$52.0_{+4.9}$**

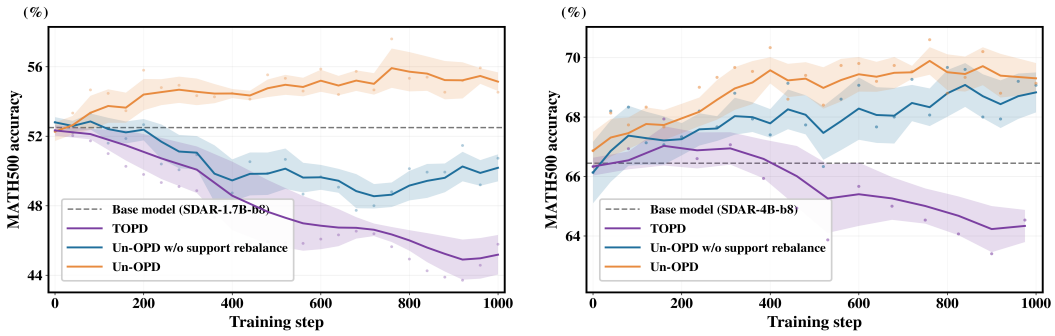
Table 2: Ablation of Un-OPD components on SDAR-4B-b8 and SDAR-1.7B-b8. We report accuracy (%) on three mathematical reasoning benchmarks and the corresponding training cost in GPU hours. Components are added cumulatively; subscripts show accuracy changes from the preceding row, computed by direct subtraction.

Method	SDAR-4B-b8				SDAR-1.7B-b8			
	MATH500	AMC2023	Olympiad	GPU hours	MATH500	AMC2023	Olympiad	GPU hours
Base	66.3	42.5	29.2	–	51.9	25.9	17.3	–
+ TOPD	$67.1_{+0.8}$	$43.6_{+1.1}$	$30.8_{+1.6}$	152	$50.9_{-1.0}$	$25.5_{-0.4}$	$17.2_{-0.1}$	104
+ context-aligned	$69.5_{+2.4}$	$44.5_{+0.9}$	$32.6_{+1.8}$	152	$51.5_{+0.6}$	$26.7_{+1.2}$	$17.3_{+0.1}$	104
+ support-rebalanced	$70.3_{+0.8}$	$45.7_{+1.2}$	$33.0_{+0.4}$	152	$54.4_{+2.9}$	$27.5_{+0.8}$	$18.9_{+1.6}$	104
+ rollout reuse	$70.1_{-0.2}$	$45.8_{+0.1}$	$32.8_{-0.2}$	72	$54.9_{+0.5}$	$27.7_{+0.2}$	$18.6_{-0.3}$	48

schedule with confidence threshold 0.9 and temperature 1.0 for evaluation unless otherwise specified. We report mean accuracy over four sampled responses per problem (avg @4), except for AMC 2023, where we use 16 responses per problem (avg @16). All models are trained for 1,000 gradient steps on an 8-GPU NVIDIA H20 server to guarantee convergence. Full training, implementation, baseline, and evaluation details are provided in Appendix B.

4.2 Main Results

Tab. 1 shows distillation performance when scaling to students with a larger block size ($b = 8$) under a teacher ($b = 4$). Across both student scales, Un-OPD outperforms TOPD on all six benchmarks and achieves the highest or tied-highest average accuracy in



(a) SDAR-1.7B-b8. (b) SDAR-4B-b8.

Figure 3: **Training dynamics on MATH500.** We compare the base model (dashed line), TOPD, Un-OPD without support rebalance, and the complete Un-OPD. Solid lines show the smoothed performance trend.

both domains. For the `SDAR-4B-b8` student, Un-OPD raises average accuracy in mathematical reasoning from 57.5% to 59.7% and in code generation from 58.7% to 63.4%. The advantage is even more striking on the smaller `SDAR-1.7B-b8` student: while TOPD suffers from severe performance degradation and consistently falls below the base model, Un-OPD successfully stabilizes training, elevating math and code averages to 44.1% and 49.0%. Overall, these results support the effectiveness of addressing optimization biases when transferring knowledge into larger-block BDLMs. Moreover, for BDLMs with the same block size (*i.e.,* 4), Un-OPD still achieves better overall performance than TOPD. For example, the mathematical reasoning accuracy increases from 46.5% to 47.0% for `SDAR-1.7B-b4`, yielding a 0.5% gain. These results demonstrate that Un-OPD readily generalizes across varying block-size configurations.

Table 3: Sensitivity to the interpolation coefficient λ on MATH500.

λ	0.2	0.4	0.6	0.8	1.0
SDAR-1.7B-b8	53.7	54.9	54.1	52.4	50.9
SDAR-4B-b8	70.0	70.1	69.8	68.7	67.1

4.3 Ablation Study

Ablation of core components. Tab. 2 ablates our proposed components on both students (`SDAR-1.7B-b8` and `SDAR-4B-b8`). Directly applying standard OPD (TOPD) yields marginal gains on `SDAR-4B-b8` and even causes performance degradation on the smaller `SDAR-1.7B-b8`, confirming the deficiency of trace-aligned distillation. Introducing **context-aligned distillation** improves all three reported benchmarks on both student scales, with MATH500 gains of 2.4% for the `SDAR-4B-b8`. However, context alignment alone leaves the weaker `SDAR-1.7B-b8` student underperforming. Adding **support-rebalanced distillation** to calibrate updates at high-support positions improves accuracy on all three reported benchmarks for the `SDAR-1.7B-b8` student (*e.g.,* a gain of 2.9% on MATH500). Finally, the **rollout reuse** mechanism reduces training overhead (from 152 to 72 GPU hours for 4B) while largely preserving accuracy. These ablations validate that each component is indispensable to achieving both superior accuracy and training efficiency.

Furthermore, Fig. 3 compares the training dynamics on MATH500 across both student scales. While standard TOPD suffers from severe late-stage degradation, Un-OPD maintains steady improvements throughout training. Crucially, omitting support rebalance causes the smaller `SDAR-1.7B-b8` student to collapse below base accuracy and limits the gains of `SDAR-4B-b8`. This confirms that support-rebalanced calibration is essential to ensure robust convergence, particularly for capacity-constrained models.

Hyper-parameter analysis of λ and N. We examine the sensitivity of Un-OPD to the interpolation coefficient λ and rollout reuse epoch number N on MATH500. For λ, Tab. 3 shows that both student scales achieve their highest reported accuracy at $\lambda = 0.4$: 54.9% for `SDAR-1.7B-b8` and 70.1% for `SDAR-4B-b8`. Increasing λ beyond 0.4 reduces accuracy on both models, supporting a moderate calibration strength. For rollout reuse, Fig. 4 compares the training dynamics of `SDAR-1.7B-b8` under $N \in \{1, 5, 10, 40\}$. Moderate reuse largely preserves performance: by the

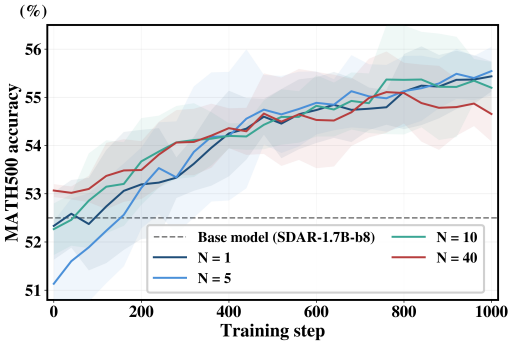


Figure 4: Training dynamics on MATH500 with different rollout reuse epochs N on SDAR-1.7B-b8. Solid lines show accuracy trajectories; the dashed line denotes the base model.

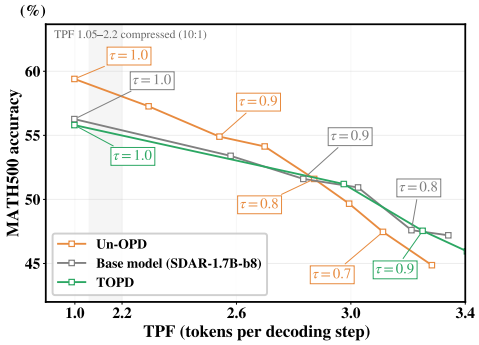


Figure 5: Accuracy versus tokens per forward pass (TPF) on MATH500 for Un-OPD and the SDAR-1.7B-b8 base model across decoding thresholds τ.

end of training, $N = 5$ and $N = 10$ achieve similar performance to $N = 1$ (the non-reused baseline). In contrast, with $N = 40$, performance declines during later training, likely due to off-policy shift from overly stale rollout trajectories. Taken together, we set $\lambda = 0.4$ and $N = 10$ to balance accuracy and rollout generation cost.

Analysis of tokens per forward pass (TPF). To evaluate the trade-off between reasoning accuracy and decoding parallelism, we sweep the dynamic decoding threshold τ on MATH500 (Fig. 5). Across high-to-moderate confidence thresholds, Un-OPD consistently establishes a superior Pareto frontier over the base model, achieving up to a $+3.3\%$ accuracy gain under conservative decoding ($\tau = 1.0$). However, as decoding becomes overly aggressive (higher TPF), this performance advantage narrows and eventually diminishes, leaving room for further improvement.

Out-of-distribution (OOD) transferability. To test cross-domain robustness, we evaluate `SDAR-4B-b8` across math and code domains (Tab. 4). After post-training on math, Un-OPD largely preserves code-generation performance, maintaining virtually identical accuracy on MBPP (53.3%) and HumanEval (68.0%). Furthermore, code-trained Un-OPD exhibits modest positive transfer to math reasoning, outperforming Base by $+0.9\%$ on MATH500 and $+1.1\%$ on AMC2023. These results confirm that Un-OPD avoids out-of-domain degradation while facilitating cross-task generalization.

Table 4: Cross-domain accuracy of SDAR-4B-b8. Arrows denote training $\rightarrow$ evaluation domains.

Method	**Code $\rightarrow$ Math**		**Math $\rightarrow$ Code**	
	MATH500	AMC2023	MBPP	HumanEval
Base	66.3	42.5	53.5	67.9
Un-OPD	67.2	43.6	53.3	68.0

5 Conclusion

In this work, we pioneer the study of scaling on-policy distillation (OPD) to block diffusion language models (BDLMs) with varying block sizes. We uncover two fundamental optimization bottlenecks: *context misalignment* induced by mismatched block boundaries, and *overconfidence collapse* triggered by biased learning dynamics under reverse-KL supervision. To overcome these challenges, we introduce **Un-OPD**, which eliminates supervisory distortions via boundary-aware step filtering and recalibrates optimization intensity across token supports. In addition, we integrate a rollout reuse mechanism that roughly halves training overhead with negligible accuracy loss. Comprehensive experiments on mathematical reasoning and code generation demonstrate that Un-OPD achieves remarkable stability and superior accuracy as well as substantial efficiency. Despite these gains, the effectiveness of Un-OPD may depend on the quality and suitability of the teacher's supervision. Furthermore, pushing this Pareto frontier toward ultra-high parallelism regimes presents a compelling avenue for future research.

AI USE STATEMENT

We used OpenAI's ChatGPT/Codex tools to support the preparation and revision of this manuscript. This assistance included drafting and revising passages, improving grammar and wording, refining paragraph structure, and maintaining consistent terminology across sections. We also used these tools to discuss the presentation of methodological and theoretical explanations, including clarifying notation, making intermediate reasoning steps explicit, and improving the connection between the proposed method and its supporting analysis. In the experimental discussion, the tools provided suggestions for experimental design and analysis and assisted with organizing comparisons and describing observed training dynamics. They also supported consistency checks across the main text, tables, figure captions, and appendices, as well as arithmetic calculations and numerical summaries based on experimental results supplied by the authors. Additional assistance involved LaTeX editing and formatting, including the presentation of equations, figures, tables, cross-references, and supplementary material. The authors checked and reviewed all AI-generated content and results included in this work, including proposed wording, methodological and theoretical explanations, interpretations of experimental observations, and numerical calculations. The authors made the final decisions about which suggestions to incorporate and take responsibility for the complete final content of this work, including all text, scientific claims, analyses, and artifacts produced with AI assistance.

ETHICS STATEMENT

This work improves the training stability and efficiency of block diffusion language models through on-policy distillation. Students trained with Un-OPD may inherit biases, factual errors, or unsafe code-generation behaviors from the teacher and training data. The reported gains on mathematical reasoning and code-generation benchmarks do not establish safety or reliability beyond the evaluated settings. More efficient generation may also lower the cost of producing misleading content or insecure code at scale. Applications of the method should therefore include task-specific safety evaluation and appropriate review of generated outputs.

REPRODUCIBILITY STATEMENT

To support reproducibility, Sec. 3 specifies the Un-OPD objective, context-aligned step selection, support-rebalanced calibration, and rollout reuse mechanism, while Algorithm 1 in Appendix A.1 presents the training procedure. Experimental settings, model configurations, and data sources are described in Sec. 4.1 and Appendix B, which also documents hardware, training hyperparameters, prompt templates, baseline configurations, decoding settings, evaluation metrics, and answer-verification procedures. For the theoretical result, Appendix D.1 provides the complete proof of Proposition 1, together with its assumptions and scope. Additional training-dynamics analyses and the procedures used to construct the corresponding smoothed curves are provided in Appendix C.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

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.

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.

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

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dopsd: On-policy self-distillation for diffusion language models. *arXiv preprint arXiv:2607.04428*, 2026.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Naibin Gu, Chenxu Yang, Qingyi Si, Chuanyu Qin, Dingyu Yao, Peng Fu, Zheng Lin, Weiping Wang, Nan Duan, and Jiaqi Wang. Co-evolving policy distillation. *arXiv preprint arXiv:2604.27083*, 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.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, Chengquan Zhang, Zhuotao Tian, Han Hu, Yi Yang, Fei Wu, and Hehe Fan. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Tianren Ma, Mu Zhang, Yibing Wang, and Qixiang Ye. Consolidating reinforcement learning for multimodal discrete diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 147950–147971, 2026.

Yixiu Mao, Yun Qu, Qi Wang, Heming Zou, and Xiangyang Ji. Rlvr without ineffective samples: Group prioritized off-policy optimization for llm reasoning. *arXiv preprint arXiv:2606.01281*, 2026.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

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. *arXiv preprint arXiv:2502.09992*, 2025.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled rl for diffusion llms emerges from a sequence-level perspective. In *International Conference on Learning Representations*, volume 2026, pp. 4941–4966, 2026.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Zheyu Fu, Liancheng Fang, Minghua He, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, et al. d-treerpo: Towards more reliable policy optimization for diffusion language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21801–21822, 2026a.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. Rlcsd: Reinforcement learning with contrastive on-policy self-distillation. *arXiv preprint arXiv:2606.11709*, 2026b.

Haolin Ren, Ziyang Huang, Chenhao Yuan, and Jun Zhao. Trace-based on-policy distillation for masked diffusion language models. *arXiv preprint arXiv:2607.16872*, 2026.

Yi Ren and Danica Sutherland. Learning dynamics of llm finetuning. In *International Conference on Learning Representations*, volume 2025, pp. 70523–70563, 2025.

Yawen Shao, Jie Xiao, Kai Zhu, Yu Liu, Hongchen Luo, Xueyang Fu, Yang Cao, Wei Zhai, and Zheng-Jun Zha. Back on track: Aligning rewards and states for reasoning in diffusion large language models. *arXiv preprint arXiv:2606.08501*, 2026.

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

Hongze Tan, Zihan Wang, Jianfei Pan, Jinghao Lin, Hao Wang, Yifan Wu, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. Gtpo and grpo-s: Token and sequence-level reward shaping with policy entropy. *arXiv preprint arXiv:2508.04349*, 2025.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *International Conference on Learning Representations*, volume 2026, pp. 150357–150385, 2026.

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, et al. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*, 2025.

Chen Wang, Lai Wei, Yanzhi Zhang, Chenyang Shao, Zedong Dan, Weiran Huang, Yuzhi Zhang, and Yue Wang. Eframe: Deeper reasoning via exploration-filter-replay reinforcement learning framework. *arXiv preprint arXiv:2506.22200*, 2025.

Jiaxuan Wang, Xuan Ouyang, Zhiyu Chen, Yulan Hu, Zheng Pan, Xin Li, and Lan-Zhe Guo. Trace: Distilling where it matters via token-routed self on-policy alignment. *arXiv preprint arXiv:2605.10194*, 2026a.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *International Conference on Learning Representations*, volume 2026, pp. 101476–101501, 2026b.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Yongliang Wu, Yizhou Zhou, Ziheng Zhou, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, et al. On the generalization of sft: A reinforcement learning perspective with reward rectification. In *International Conference on Learning Representations*, volume 2026, pp. 27550–27571, 2026a.

Zirui Wu, Lin Zheng, Jiacheng Ye, Shansan Gong, Xueliang Zhao, Yansong Feng, Wei Bi, and Lingpeng Kong. Dreamreasoner-8b: Block-size curriculum learning for diffusion reasoning models. *arXiv preprint arXiv:2606.19257*, 2026b.

Yan Xie, Sijie Zhu, Tiansheng Wen, Bo Chen, and Yifei Wang. On the position bias of on-policy distillation. *arXiv preprint arXiv:2606.22600*, 2026.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Tip: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026a.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026b.

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.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12023–12040, 2025.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. In *International Conference on Learning Representations*, volume 2026, pp. 12810–12846, 2026.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D Lyng, Sanjit Singh Batra, and Robert E Tillman. Fast and effective on-policy distillation from reasoning prefixes. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25553–25569, 2026.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. Rlep: Reinforcement learning with experience replay for llm reasoning. *arXiv preprint arXiv:2507.07451*, 2025.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:56729–56762, 2026.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

Appendix Contents

A Framework Details 16

A.1 Algorithm Details 16

B Experimental Details 17

B.1 Training Configuration 17

B.2 Prompt Template 18

B.3 Baseline Configurations 18

B.4 Evaluation Benchmarks 19

C Further Experimental Analysis 19

C.1 Comparison under Different Context Alignment Regimes 19

C.2 Comparison under a Static Decoding Schedule 21

C.3 Comparison with Symmetric Block Sizes 21

C.4 Extension to Larger Block Sizes 21

C.5 Response Length Analysis 22

D Proofs 22

D.1 Proof of Proposition 1 23

E Case Study 25

A Framework Details

A.1 Algorithm Details

Algorithm 1 summarizes the Un-OPD training loop. Following TOPD (Ren et al., 2026), each rollout round comprises three stages: sampling prompts and responses from the current student, constructing training examples from the recorded denoising traces, and optimizing the student with frozen-teacher supervision. Each sample pairs a committed token position with its decoding state, attaching supervision to decisions made along the student's actual generation process. We reconstruct the visible context under each model's block schedule and retain only decisions satisfying $s_{\mathrm{T},k} = s_{\mathrm{S},k}$, so teacher and student predictions are compared under the same context. At retained positions with $\rho_{k,i} > 1$, teacher logits are interpolated with detached student logits before computing the reverse KL. The resulting trace batch is reused for N optimization epochs to reduce rollout sampling overhead.

Algorithm 1: Un-OPD: Context-Aligned Distillation with Support Rebalance

Require: Student π_θ; frozen teacher π_{T}; prompt set $\mathcal{D}$; prompt batch size M; rollouts per prompt G; update minibatch size m (traces); student/teacher block sizes $B_{\mathrm{S}}, B_{\mathrm{T}}$; student decoding schedule; maximum response length $L_{\max}$; calibration coefficient λ; replay epochs N; learning rate η.

for *each rollout round* **do**
  Sample a prompt batch $\mathcal{B} \sim \mathcal{D}$ with $|\mathcal{B}| = M$; initialize buffer $\mathcal{R} \leftarrow \varnothing$
  `// Collect student traces and select context-aligned decisions`
  for each $q \in \mathcal{B}$ *and* $g = 1, \ldots, G$ **do**
    Initialize masked response state s_0; set $k \leftarrow 0$
    while *the response is unfinished and below* $L_{\max}$ **do**
      Sample committed decisions $a_k \sim \pi_\theta(\cdot \mid q, s_k)$ under the decoding schedule
      $s_{k+1} \leftarrow \mathrm{unmask}(s_k, a_k)$; $k \leftarrow k + 1$
    end
    Store trace $\xi = (s_0, a_0, \ldots, a_{K-1}, s_K)$ with $K = k$
    Reconstruct visible states $s_{\mathrm{S},k}, s_{\mathrm{T},k}$ using $B_{\mathrm{S}}, B_{\mathrm{T}}$
    $\mathcal{A}_{\mathrm{align}}(\xi) \leftarrow \{(k, i) : (i, x_i) \in a_k,\ 0 \le k < K,\ s_{\mathrm{T},k} = s_{\mathrm{S},k}\}$
    if $\mathcal{A}_{\mathrm{align}}(\xi) \neq \varnothing$ **then**
      Add $(q, \xi, \mathcal{A}_{\mathrm{align}}(\xi))$ to $\mathcal{R}$
    end
  end
  if $\mathcal{R} = \varnothing$ **then**
    Repeat rollout sampling and alignment filtering until $\mathcal{R} \neq \varnothing$
  end
  `// Replay fixed traces with support-aware teacher supervision`
  for $n = 1, \ldots, N$ **do**
    Shuffle $\mathcal{R}$ and split it into minibatches of at most m traces
    for each *minibatch* $\mathcal{M}$ **do**
      for each $(q, \xi, \mathcal{A}_{\mathrm{align}}(\xi)) \in \mathcal{M}$ **do**
        for each $(k, i) \in \mathcal{A}_{\mathrm{align}}(\xi)$ **do**
          Compute student logits $z_{\mathrm{S},k,i}$ at $(q, s_{\mathrm{S},k}, i)$ and frozen teacher logits $z_{\mathrm{T},k,i}$ at $(q, s_{\mathrm{T},k}, i)$
          $p_{\mathrm{S},k,i} \leftarrow \mathrm{softmax}(z_{\mathrm{S},k,i})$; $p_{\mathrm{T},k,i} \leftarrow \mathrm{softmax}(z_{\mathrm{T},k,i})$
          $h_{k,i} \leftarrow \arg\max_v p_{\mathrm{S},k,i}(v)$; $\rho_{k,i} \leftarrow p_{\mathrm{T},k,i}(h_{k,i}) / p_{\mathrm{S},k,i}(h_{k,i})$
          $\widetilde{z}_{\mathrm{T},k,i} \leftarrow z_{\mathrm{T},k,i}$
          if $\rho_{k,i} > 1$ **then**
            $\widetilde{z}_{\mathrm{T},k,i} \leftarrow \lambda z_{\mathrm{T},k,i} + (1 - \lambda)\,\mathrm{sg}[z_{\mathrm{S},k,i}]$
          end
          $\ell_{k,i} \leftarrow D_{\mathrm{KL}}(p_{\mathrm{S},k,i} \,\|\, \mathrm{sg}[\mathrm{softmax}(\widetilde{z}_{\mathrm{T},k,i})])$
        end
        $\mathcal{L}_\xi \leftarrow |\mathcal{A}_{\mathrm{align}}(\xi)|^{-1} \sum_{(k,i) \in \mathcal{A}_{\mathrm{align}}(\xi)} \ell_{k,i}$
      end
      $\mathcal{L}_{\mathcal{M}} \leftarrow |\mathcal{M}|^{-1} \sum_{\xi \in \mathcal{M}} \mathcal{L}_\xi$; $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\mathcal{M}}$
    end
  end
end

Rollout states and committed decisions are held fixed during replay, while student predictions, top-1 tokens, and calibration targets are recomputed at each update. No gradient passes through the teacher, the support-based selection, or the calibrated targets. Traces without aligned decisions are filtered. If all traces in a batch are filtered out, we repeat rollout sampling and alignment filtering until the buffer is nonempty, then proceed with student updates. Averaging within each trace before averaging across traces follows Eq. 7; the parameter update is written as gradient descent for clarity.

B Experimental Details

This section documents the training configuration, prompt templates, baseline settings, and evaluation benchmarks for our experiments on `SDAR` series models.

B.1 Training Configuration

Tab. 5 summarizes the detailed experimental setup for Un-OPD, covering model architectures, rollout sampling, optimization, and datasets. Overall configurations follow TOPD (Ren et al., 2026), with key hyperparameters for context-aligned step selection, support rebalance (λ), and rollout replay (N) explicitly detailed. The paired model scales and block sizes are consistent with those in Tab. 1, while the specific effects of λ and N are thoroughly investigated in Sec. 4.3.

Table 5: Training configuration for Un-OPD on SDAR students.

Category	**Value**
Model and Initialization	
Student models	`SDAR-1.7B` and `SDAR-4B`
Teacher model	`TraDo-8B`
Hardware configuration	Single node with 8 NVIDIA H20 GPUs
Precision	bfloat16; TF32 enabled
Gradient checkpointing	Enabled (whole layer)
Rollout (On-Policy Sampling)	
Tasks (prompts) per rollout round	64
Responses per prompt	1
Sampling temperature	1.0
Maximum generation length	2,000 tokens
Student / teacher block sizes	Per configuration in Tab. 1
Denoising steps per block	student block size
Remasking strategy	Low-confidence dynamic ($\tau = 0.9$)
Top-p	1.0
Training and Optimization	
Optimizer	AdamW
Learning rate	2×10^{-7} (constant)
Weight decay	0.0
Effective optimization batch size	32 (4/GPU $\times$ 8 $\times$ 1 accum.)
Gradient clipping (max norm)	1.0
Un-OPD-Specific Settings	
Loss type	Full-vocabulary reverse KL
Teacher status	Frozen
Interpolation coefficient λ	0.4
Rollout replay epochs N	10
Data	
Training dataset	8K math problems following TraceRL (Wang et al., 2026b)
Maximum prompt length	784 tokens

Training infrastructure and execution. We train all Un-OPD models on a single node equipped with eight NVIDIA H20 GPUs. Each rollout cycle samples a batch of 64 prompts, followed by N optimization epochs over the retained, context-aligned traces. During rollout replay, the teacher

parameters and rollout trajectories remain frozen, whereas student predictions and the resulting calibrated targets are dynamically recomputed at each gradient update, as detailed in Algorithm 1.

B.2 Prompt Template

Fig. 6 presents the math prompt template for `SDAR` models. The placeholder {{problem}} is replaced with the problem statement. The instruction requests step-by-step reasoning and a final answer enclosed in \boxed{}, followed by the assistant prefix from which generation begins.

Math prompt template

SDAR

```
<|im_start|>user
{{problem}}
Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>
<|im_start|>assistant
```

Figure 6: Math prompt template for SDAR, with {{problem}} denoting the input problem.

For code generation, Fig. 7 shows the template with an in-context prefix, {{icl_prefix}}, prepended to the problem statement. The instruction requires the response code to appear in a single Python code block.

Code prompt template

SDAR

```
<|im_start|>user
{{icl_prefix}}{{problem}}
Place your code within a single Python code block `python `.  Do
not include more than one code block.  <|im_end|>
<|im_start|>assistant
```

Figure 7: Code prompt template for SDAR. The placeholders denote the in-context prefix and the input problem.

B.3 Baseline Configurations

We compare Un-OPD with trace-based reinforcement learning (**TraceRL** (Wang et al., 2026b)), supervised fine-tuning, and trace-aligned on-policy distillation (**TOPD**). Student scales and block sizes follow the corresponding groups in Tab. 1.

TraceRL. We train each student model using the TraceRL codebase (Wang et al., 2026b). TraceRL combines the denoising-trace formulation with group-relative policy optimization (Shao et al., 2024). For each prompt, it samples a group of $G = 16$ responses and computes relative advantages from their binary correctness rewards. Each complete response receives a reward of 1 if it solves the problem correctly and 0 otherwise. The update uses token log-probabilities evaluated at the recorded decoding states, with old-policy log-probabilities held fixed throughout optimization of the rollout batch. A clipped policy objective limits policy changes, while a KL penalty regularizes the student toward a frozen reference policy. We use a clipping threshold of $\epsilon = 0.2$, a KL coefficient of $\beta = 0.01$, a learning rate of 1×10^{-6} with cosine decay, and an effective optimization batch size of 16. Advantages are computed within each response group, without a learned value model for simplicity.

Semi-autoregressive SFT. The SFT baseline fine-tunes the student on offline teacher trajectories using the block-diffusion masking objective established by `SDAR` (Cheng et al., 2026) and TraceRL (Wang et al., 2026b). Specifically, fully generated preceding blocks serve as causal context, while training optimizes the cross-entropy loss over masked tokens within the supervised block. Following the SFT recipe in TraceRL, the masking ratio is drawn uniformly from $[0.1, 0.9]$, and the block size is configured to match the student under evaluation. Models are trained for one epoch with a learning rate of 1×10^{-5}. For mathematical reasoning, the training corpus consists of dynamic decoding responses sampled by the `TraDo-8B` teacher on the same 8K math problems, following the training setup of TraceRL (Wang et al., 2026b). Likewise, code training trajectories are sampled from the AceCoder subset using the same teacher model.

TOPD. TOPD (Ren et al., 2026) serves as the standard trace-aligned on-policy distillation baseline. It samples rollouts from the student policy and supervises committed token decisions along intermediate denoising traces. At each supervised position, it minimizes the full-vocabulary reverse-KL divergence against a frozen teacher evaluated directly on the student's decoding states. TOPD directly applies standard OPD to BDLMs without accounting for how supervision at different denoising steps affects student optimization. For a fair comparison, we reproduce TOPD across different student models using the configurations reported in the original paper.

B.4 Evaluation Benchmarks

We evaluate mathematical reasoning and code generation across six widely adopted benchmarks, as summarized in Tab. 1.

Mathematical reasoning. We assess problem-solving capabilities across a diverse spectrum of difficulty using four benchmarks: (1) **GSM8K** evaluates multi-step grade-school arithmetic reasoning; (2) **MATH-500** comprises a 500-problem subset of the challenging MATH benchmark across algebra, geometry, and number theory; (3) **AMC 2023** comprises 40 competition-level problems drawn from the 2023 American Mathematics Competitions; and (4) **OlympiadBench** tests advanced, Olympiad-level mathematical reasoning. Together, these benchmarks span elementary arithmetic to elite competition mathematics.

Code generation. We employ **HumanEval** and **MBPP** to evaluate standalone Python program synthesis from docstrings and natural-language problem descriptions. For both benchmarks, correctness is strictly assessed through the functional execution of official unit tests.

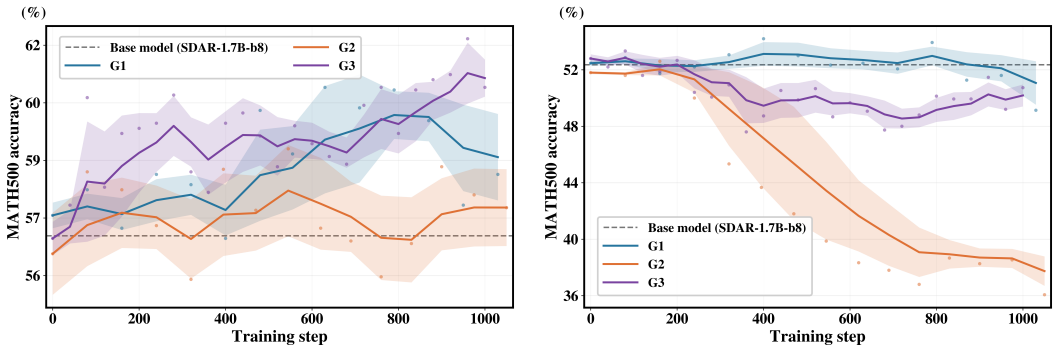
Sampling and evaluation metrics. We evaluate BDLMs using a dynamic decoding schedule (confidence threshold $\tau = 0.9$), with a temperature of 1.0 and top-$p = 1.0$, under their respective block sizes unless otherwise specified. We report the average accuracy metric, $\text{avg}\,@k$, computed as the mean instance-level pass rate over k sampled responses per problem. We sample four responses per problem and report $\text{avg}\,@4$ for all benchmarks except AMC 2023, for which we sample 16 responses per problem and report $\text{avg}\,@16$ to mitigate sampling variance arising from its compact 40-problem test set.

Answer verification. For all mathematical reasoning benchmarks (GSM8K, MATH-500, AMC 2023, and OlympiadBench), we extract the final answer from `\boxed{}` and verify its mathematical equivalence to the reference answer via symbolic and numerical normalization. For MBPP and HumanEval, we execute the extracted code directly in a sandboxed Python environment, counting a response as correct only if it passes all assertion test cases.

C Further Experimental Analysis

C.1 Comparison under Different Context Alignment Regimes

Here we investigate how the different context regimes defined in Sec. 3.2 affect student model performance. Complementing the teacher-support and confidence analysis for `SDAR-4B-b8` in Fig. 1, we train separate OPD runs using only the decoding steps from each regime: G_1, where the student sees more context than the teacher; G_2, where the teacher sees more context than the student; and G_3, where their visible contexts are aligned. Fig. 8 compares their MATH500 accuracy under static and dynamic decoding. In Fig. 8, scatter points denote raw evaluation accuracies at individual check-

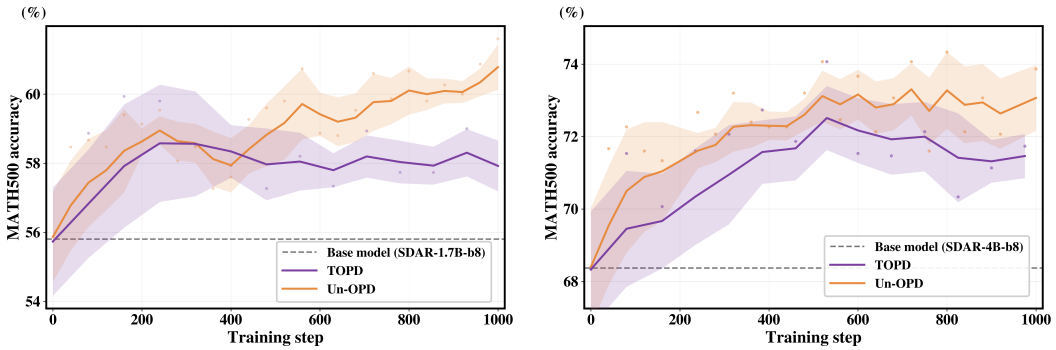


(a) Static decoding. (b) Dynamic decoding.

Figure 8: **Training dynamics under different context alignment regimes.** MATH500 accuracy for `SDAR-1.7B-b8` trained with OPD restricted to G_1 (blue), G_2 (orange), or G_3 (purple), as defined in Sec. 3.2. Points show checkpoint accuracies and solid curves show smoothed trends. Gray dashed lines indicate the base-model accuracy under each decoding schedule.

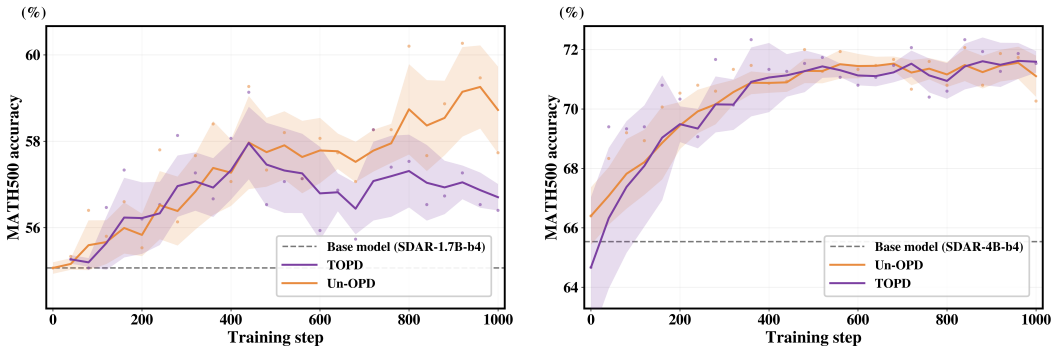
points, while solid curves trace exponential moving averages (EMA, $\alpha = 0.35$). The surrounding shaded bands represent ± 1 local standard deviation computed over a backward sliding window of size five (including the current and up to four prior checkpoints). This identical visualization scheme is consistently adopted in Figs. 9, 10, and 11.

In Fig. 8(a), training on G_3 obtains the strongest distillation performance, with the smoothed accuracy reaching approximately 61%. The G_1 run also improves over the base model, but loses part of its gains near the end of training. We attribute this to the lower teacher support inherent to G_1 steps, which penalizes confident student predictions even when correct, thereby inducing underconfidence and curtailing the model's ultimate decoding potential. Meanwhile, training solely on G_2 achieves little gain over the baseline throughout training. We argue that G_2 steps inject miscalibrated, overly aggressive supervision, rewarding high confidence on inaccurate predictions and consequently degrading the model's true decoding capacity. These results is consistent with the teacher signals biases identified in Sec. 3.2: aligned contexts provide more balanced teacher support, whereas either direction of context mismatch can reduce the effectiveness of distillation. In Fig. 8(b), we find that G_1 marginally outperforms the other two regimes. We attribute this to the student's lower confidence under G_1, which favors dynamic decoding by encouraging overly cautious unmasking schedules. However, this superficial boost comes at the cost of the model's fundamental decoding capacity, as demonstrated by its degraded static performance. Consequently, in our context-aligned distillation, we intentionally restrict OPD to aligned steps (G_3) to preserve the model's performance.



(a) SDAR-1.7B-b8. (b) SDAR-4B-b8.

Figure 9: **Training dynamics on MATH500 under static decoding.** Purple and orange curves correspond to TOPD and Un-OPD, respectively.



(a) SDAR-1.7B-b4. (b) SDAR-4B-b4.

Figure 10: **Training dynamics with matched teacher and student block sizes.** MATH500 accuracy under dynamic decoding with teacher and student being BDLMs with identical block sizes (4). Orange and purple curves correspond to Un-OPD and TOPD, respectively.

C.2 Comparison under a Static Decoding Schedule

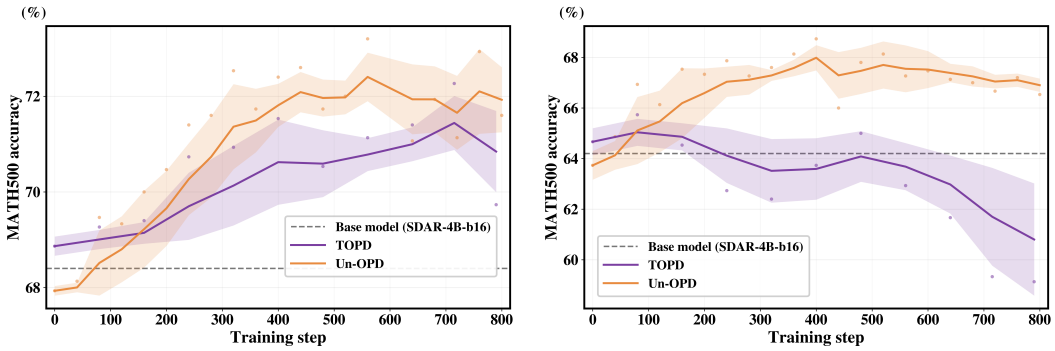
Different from the dynamic decoding schedule, the static schedule unmasks exactly one token per step, serving as an indicator of intrinsic model capability. To complement the dynamic-decoding results in Fig. 3, we evaluate TOPD and Un-OPD under static decoding throughout training (Fig. 9). For `SDAR-1.7B-b8`, both methods improve early on, but TOPD quickly plateaus around 58%, whereas Un-OPD continues climbing to 60.8% at step 1,000 (+2.9% over TOPD). For `SDAR-4B-b8`, Un-OPD stabilizes at approximately 73.0% around step 500, while TOPD degrades after peaking. Collectively, these results confirm that Un-OPD not only excels under dynamic decoding schedule, but also substantially elevates the model's intrinsic performance upper bound.

C.3 Comparison with Symmetric Block Sizes

We further investigate training dynamics under a symmetric configuration where both the teacher and student adopt an identical block size of 4 (Fig. 10). For `SDAR-1.7B-b4`, both methods improve initially, but their trajectories markedly diverge around step 450: TOPD undergoes late-stage degradation, eroding its earlier gains, whereas Un-OPD stably preserves accuracy and improves further toward the end of training. In contrast, for `SDAR-4B-b4`, both methods follow comparable trajectories, plateauing around 71%–72% without a decisive gap. These observations reveal that block-size symmetry alone does not eliminate the training instability in capacity-constrained students. Consequently, Un-OPD's stabilizing mechanism proves crucial for weaker students, while maintaining competitive parity on larger models under symmetric regimes.

C.4 Extension to Larger Block Sizes

To assess whether the benefits of Un-OPD extend beyond the block sizes studied in the main experiments, we increase the student block size to 16 and compare Un-OPD with TOPD on `SDAR-4B-b16` (`Trado-8b-b4` as teacher). Fig. 11 reports MATH500 accuracy throughout training under both static and dynamic decoding. Under static decoding, Un-OPD reaches approximately 72% and maintains a lead over TOPD (Fig. 11(a)). Under dynamic decoding, it remains near 67% late in training, while TOPD declines toward 61% (Fig. 11(b)). These results show that Un-OPD remains effective at a student block size of 16, improving attainable accuracy and stability under confidence-based decoding.



(a) Static decoding.

(b) Dynamic decoding.

Figure 11: **Scaling distillation to a student block size of 16.**

C.5 Response Length Analysis

We additionally track generated response length for `SDAR-1.7B` with block sizes $b \in \{4, 8\}$ to examine whether training is accompanied by a sustained increase in the output-token budget. Fig. 12 plots the average response length on MATH500 under dynamic decoding across training checkpoints. Length is measured in generated response tokens, excluding the prompt and padding.

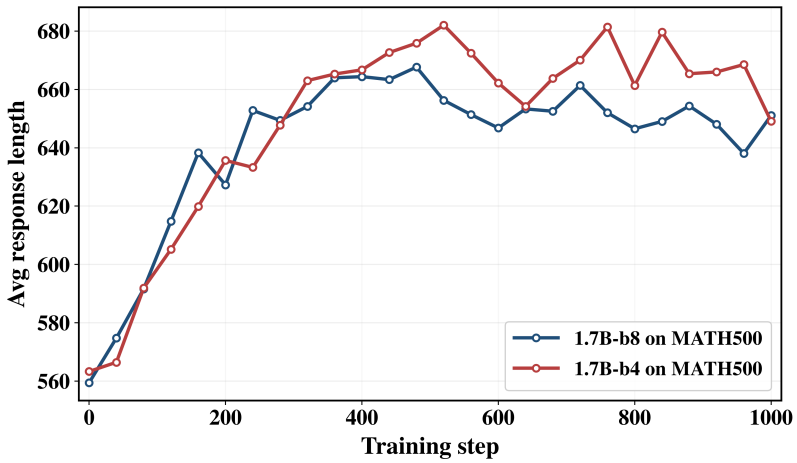


Figure 12: **Response length over training under dynamic decoding.** Average generated response length on MATH500 for SDAR-1.7B with block sizes $b = 8$ (blue) and $b = 4$ (red). Markers indicate the checkpoint averages.

Both trajectories rise from approximately 560 tokens at initialization to around 650–670 tokens. Thereafter, average response length fluctuates within a comparatively narrow range. The $b = 4$ student generally produces somewhat longer responses than the $b = 8$ student. The initial increase in response length should be considered when interpreting early accuracy gains. The subsequent plateau suggests that later training is not accompanied by an uncontrolled expansion of generated responses.

D Proofs

We prove Proposition 1 using the student's top-1 token and the conditional tail KL defined in Sec. 3.3.

D.1 Proof of Proposition 1

Top-1 token and conditional tail distributions. Fix a prediction position i, its context, and a teacher distribution with positive token probabilities. Let $p_{\mathrm{S},i} = \mathrm{softmax}(z_{\mathrm{S},i})$ and fix the student's current top-1 token $h_i = \arg\max_v p_{\mathrm{S},i}(v)$ when evaluating the update. Following the main text, write $a_i = p_{\mathrm{S},i}(h_i)$, $b_i = p_{\mathrm{T},i}(h_i)$, and $\rho_i = b_i/a_i$. For a vocabulary with at least two tokens and finite student logits, $0 < a_i, b_i < 1$. The tail consists of all vocabulary tokens other than h_i at this position. Conditioning each distribution on this set gives

$$\bar{p}_{\mathrm{S},i}(v) = \frac{p_{\mathrm{S},i}(v)}{1-a_i}, \qquad \bar{p}_{\mathrm{T},i}(v) = \frac{p_{\mathrm{T},i}(v)}{1-b_i}, \qquad v \neq h_i.$$

Both conditional distributions sum to one, so their reverse KL is nonnegative:

$$D_i^{\mathrm{tail}} = D_{\mathrm{KL}}(\bar{p}_{\mathrm{S},i} \| \bar{p}_{\mathrm{T},i}) = \sum_{v \neq h_i} \bar{p}_{\mathrm{S},i}(v) \log \frac{\bar{p}_{\mathrm{S},i}(v)}{\bar{p}_{\mathrm{T},i}(v)} \geq 0.$$

This definition uses the student's top-1 token to partition both distributions; it does not require the teacher to have the same top-1 token.

Reverse-KL decomposition. Substituting $p_{\mathrm{S},i}(v) = (1-a_i)\bar{p}_{\mathrm{S},i}(v)$ and $p_{\mathrm{T},i}(v) = (1-b_i)\bar{p}_{\mathrm{T},i}(v)$ for $v \neq h_i$ into Eq. 2 yields

$$\mathcal{L}_i = a_i \log \frac{a_i}{b_i} + (1-a_i) \log \frac{1-a_i}{1-b_i} + (1-a_i) D_i^{\mathrm{tail}}. \tag{8}$$

The first two terms together form the Bernoulli KL between the top-1 probabilities a_i and b_i. The final term measures the discrepancy within the conditional tail distributions, weighted by the student's total tail probability $1 - a_i$.

Top-1 logit update. For a fixed teacher, the softmax derivative

$$\frac{\partial p_{\mathrm{S},i}(u)}{\partial z_{\mathrm{S},i}(v)} = p_{\mathrm{S},i}(u) \left[\mathbf{1}\{u = v\} - p_{\mathrm{S},i}(v)\right]$$

gives the full-vocabulary reverse-KL gradient

$$\begin{aligned} \frac{\partial \mathcal{L}_i}{\partial z_{\mathrm{S},i}(v)} &= \sum_{u \in \mathcal{V}} \left[\log \frac{p_{\mathrm{S},i}(u)}{p_{\mathrm{T},i}(u)} + 1\right] \frac{\partial p_{\mathrm{S},i}(u)}{\partial z_{\mathrm{S},i}(v)} \\ &= p_{\mathrm{S},i}(v) \left[\log \frac{p_{\mathrm{S},i}(v)}{p_{\mathrm{T},i}(v)} - \mathcal{L}_i\right]. \end{aligned} \tag{9}$$

Evaluating this identity at $v = h_i$ and using Eq. 8, we obtain

$$\begin{aligned} \frac{\partial \mathcal{L}_i}{\partial z_{\mathrm{S},i}(h_i)} &= a_i \left[\log \frac{a_i}{b_i} - \mathcal{L}_i\right] \\ &= -a_i(1-a_i) \left[\log \frac{b_i(1-a_i)}{a_i(1-b_i)} + D_i^{\mathrm{tail}}\right]. \end{aligned} \tag{10}$$

Multiplying by $-\eta$ gives

$$\Delta z_{\mathrm{S},i}(h_i) = \eta a_i(1-a_i) \left[\log \frac{b_i(1-a_i)}{a_i(1-b_i)} + D_i^{\mathrm{tail}}\right],$$

which is exactly Eq. 5. □

Implications for the update direction. If $\rho_i > 1$, then $b_i > a_i$ and the log-odds term is strictly positive. Since $D_i^{\mathrm{tail}} \geq 0$, the top-1 logit increases. If $\rho_i < 1$, the log-odds term is negative, and the top-1 logit decreases if and only if

$$D_i^{\mathrm{tail}} < \log \frac{a_i(1-b_i)}{b_i(1-a_i)}.$$

Equality gives a zero update, while a larger conditional tail KL produces an upward update. When $\rho_i = 1$, the update reduces to $\eta a_i(1-a_i) D_i^{\mathrm{tail}}$, which is positive whenever the conditional tail distributions differ. Thus, conditional tail mismatch reinforces upward updates at high-support positions

and can weaken or reverse downward updates at low-support positions, as discussed in the main text.

The result concerns a direct gradient step on the logits at one fixed context. Changes in normalized top-1 probability and entropy also depend on updates to the other logits; training through shared model parameters additionally couples different positions. The proposition therefore characterizes a local mechanism consistent with the observed asymmetry, rather than guaranteeing a particular confidence trajectory during training.

E CASE STUDY

We compare Un-OPD and TOPD responses to the same MATH500 problem. Case 1 presents the Un-OPD response, and Case 2 presents the TOPD response. Each includes the problem and reference answer. The responses are reproduced below with formatting adjusted for typesetting.

Case 1: Un-OPD on MATH500 (ground truth $2k + 2$)

Problem. Simplify $(-k + 4) + (-2 + 3k)$.

Un-OPD output ($2k + 2$).
To simplify the expression $(-k + 4) + (-2 + 3k)$, we will follow these steps:
1. Combine like terms.
First, let's rewrite the expression:

$$(- + 4) + (- + 3k)$$

Next, distribute the negative sign in the second term:

$$-k + 4 + -2 + 3k$$

Now, combine the like terms:
- Combine the k terms: $-k + 3k = 2k$
- Combine the constant constant terms: $4 - 2 = 2$

Thus, the simplified expression is:

$$2k + 2$$

Therefore, the simplified form of the expression $(-k + 4) + (-2 + 3k)$ is:

$$\boxed{2k + 2}$$

Case 2: TOPD on MATH500 (ground truth $2k + 2$)

Problem. Simplify $(-k + 4) + (-2 + 3k)$.

TOPD output ($4k + 2$).
To simplify the expression $(-k + 4) + (-2 + 3k)$, we start by removing the parentheses and combining like terms.
First, distribute the negative signs in the second parentheses:

$$(-k + 4) + (-2 + 3k) = -k + 4 - 2 + 3k$$

Next, combine the like terms. Group the k terms together and the constant terms together:

$$k + 33 + 4 - 2$$

Combine the constants:

$$4 - 2 = 2$$

Now, combine the k terms:

$$k + 3k = 4k$$

Putting it all together, we get:

$$4k + 2$$

Thus, the simplified form is:

$$\boxed{4k + 2}$$