LightningRL: Breaking the Accuracy–Parallelism Trade-off of Block-wise dLLMs via Reinforcement Learning

Anonymous Authors[1]

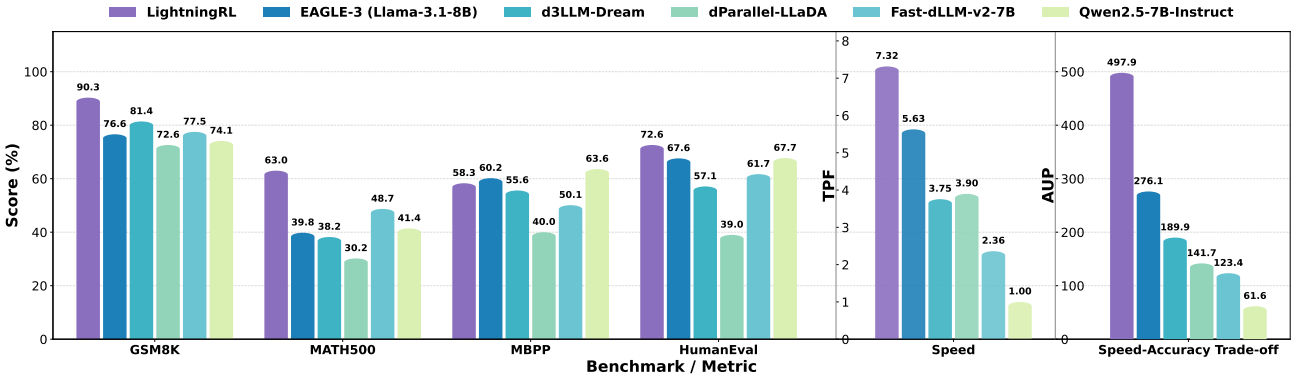


Figure 1. **Evaluation results of LightningRL and the baselines.** LightningRL achieves superior accuracy on math and code benchmarks while accelerating parallel decoding to an average of 7.32 tokens per forward (TPF) and 497.9 accuracy under parallelism (AUP) (Qian et al., 2026), significantly outperforming baselines including Eagle-3 (Li et al., 2025) and Fast-dLLM-v2 (Wu et al., 2025a).

Abstract

Diffusion Large Language Models (dLLMs) enable parallel token generation, and their block-wise variants have attracted significant attention. However, existing dLLMs usually exhibit an accuracy–parallelism trade-off, where raising tokens per forward (TPF) via aggressive parallel decoding often degrades task accuracy. To address this, we suggest developing a post-training approach to directly optimize the speed–quality frontier of pre-trained dLLMs. Conceptually, we do not require the model to decode aggressively along all sampling trajectories, but rather to find several highly parallelizable ones that can yield correct results. To this end, we resort to a reinforcement learning paradigm, i.e., LightningRL, to optimize rewards regarding both the final accuracy and inference parallelism. LightningRL follows the Group Relative Policy Optimization (GRPO) framework, with further improvements for dLLMs: 1) stabilized training via per-reward decoupled normalization, 2) token-level negative log-likelihood (NLL) loss on correct trajectories for regularization, and 3) improved training efficiency through dynamic sampling with TPF-aware filtering. Across maths and code tasks, LightningRL consistently advances the Pareto frontier, maintaining competitive accuracy while increasing parallelism to an average TPF of 7.32 (up to 11.10 on MBPP).

1. Introduction

Diffusion Large Language Models (dLLMs) are a promising alternative to autoregressive (AR) decoders for high-throughput generation (Li et al., 2022; Sahoo et al., 2024a; Gulrajani & Hashimoto, 2023; Gong et al., 2023; Sahoo et al., 2024b; Nie et al., 2025; Ye et al., 2025). They frame the generation as iterative denoising of a corrupted token sequence, enabling bidirectional context aggregation and parallel refinement over many positions. However, vanilla dLLMs can suffer from the incompatibility with KV cache mechanisms and fixed generation length.

Block-wise dLLMs (Arriola et al., 2025) address these by bridging AR and diffusion LLMs. They generate blocks of text tokens sequentially to ensure long-range coherence (AR-like), while simultaneously refining all tokens within each

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

block via diffusion to unlock intra-block parallelism. This approach mitigates the inefficiency of independent token sampling in pure diffusion and the lack of parallelism in AR. In practice, representative works construct block-wise dLLMs by adapting pretrained AR models for block-wise denoising (Cheng et al., 2025; Wu et al., 2025a).

However, existing block-wise dLLMs still suffer from a severe accuracy–parallelism trade-off (Qian et al., 2026), where raising tokens per forward (TPF) via aggressive parallel decoding often degrades task accuracy. Despite training-free sampling strategies (Wu et al., 2025b; Xu et al., 2025) and distillation-based approaches (Wang et al., 2025a; Chen et al., 2025) can boost the TPF and hence the tokens per second (TPS) during inference, it typically comes at the cost of substantial degradation in generation quality. As a result, existing dLLMs are still inferior to popular acceleration approaches to AR LLMs like speculative decoding (Leviathan et al., 2023; Li et al., 2025; Kou et al., 2024) when simultaneously considering speed and quality.

To address this, we advocate a post-training for pre-trained dLLMs that directly optimizes the speed–quality frontier. Our core insight is that we do not require the model to aggressively decode all possible paths; instead, we merely need the model to reliably navigate the specific subspace of trajectories that are both highly parallelizable and accurate. We formulate this objective as a reinforcement learning (RL) problem, using supervision from outcome accuracy and overall TPF to shape the model's probability mass. We implement the resulting **LightningRL** upon the widely-used Group Relative Policy Optimization (GRPO) framework (Shao et al., 2024).

LightningRL makes necessary modifications to GRPO, including 1) *Decoupled Reward Normalization*, which addresses the scale discrepancy between accuracy and TPF rewards via independent normalization to ensure stable multi-objective optimization, 2) *Likelihood-Anchored Regularization*, where a token-level negative log-likelihood (NLL) objective computed on correct trajectories is leveraged to mitigate reward hacking and stabilize updates, and 3) *TPF-aware Filtering*, which selects prompts with sampling trajectories of diverse levels of parallelism to maintain distinct learning signals and improve sample efficiency.

Empirically, we perform LightningRL post-training on the representative block-wise dLLMs SDAR (Cheng et al., 2025) and validate on a comprehensive suite of math and code benchmarks. We report accuracy, TPF (as a measure of parallelism), and accuracy under parallelism (AUP) (Qian et al., 2026), which summarizes the speed–quality trade-off under parallel decoding. The results highlight a superior trade-off profile: our LightningRL-8B model with a block size of 32, tuned from the SDAR-8B, achieves 497.9 AUP with an average speed of 7.32 TPF (up to 11.10 TPF on MBPP). This performance substantially surpasses established acceleration baselines such as d3LLM (Qian et al., 2026) and EAGLE-3 (Li et al., 2025), demonstrating the ability of LightningRL to effectively break the accuracy–parallelism bottleneck of dLLMs.

2. Preliminaries

2.1. Diffusion Large Language Models (dLLMs)

Given an input prompt q, a dLLM generates the response as a Markov Decision Process (MDP) (Li et al., 2022). Ideally, this process produces a trajectory of intermediate states $\{x_0, x_1, \ldots, x_T\}$, where each $x_t \in (\mathcal{V} \cup \{[\text{MASK}]\})^L$ represents the partially decoded sequence at step t. Here, $\mathcal{V}$ denotes the vocabulary, L is the sequence length, and x_0 is the initial fully masked state.

Let $p_\theta(\cdot \mid x_t, q)$ denote the generation distribution parameterized by the model, which effectively serves as the transition policy. The generation proceeds by iteratively sampling the next state based on the current context:

$$x_{t+1} \sim p_\theta(x_{t+1} \mid x_t, q). \tag{1}$$

This aligns with the Markov property, where the next state depends only on the current state x_t and the condition q.

Confidence-driven Decoding (Wu et al., 2025b; Wang et al., 2025a) specifies the transition dynamics by allowing multiple tokens in x_t to be accepted (decoded) in a single iteration if their prediction confidence exceeds a threshold, while rejected tokens are reset to [MASK] in x_{t+1}.

Block Diffusion Vanilla dLLMs suffer from the incompatibility with KV cache mechanisms and fixed generation length (Nie et al., 2025; Ye et al., 2025). Block-wise dLLMs (Arriola et al., 2025; Cheng et al., 2025) address these by partitioning the sequence x into B contiguous blocks $\{x^1, \ldots, x^B\}$, where blocks are generated sequentially while tokens within each block are decoded in parallel. Block-wise dLLMs can be efficiently adapted from pre-trained AR LLMs, with SDAR (Cheng et al., 2025) and Fast-dLLM-v2 (Wu et al., 2025a) as popular examples.

2.2. GRPO for dLLMs

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) has demonstrated remarkable effectiveness for the reinforcement learning (RL) of AR LLMs. Prior works have successfully adapted this paradigm to the non-autoregressive decoding of dLLMs (Wang et al., 2025b; Zhu et al., 2026). To align the RL objective with the dLLM generation process, we explicitly map the policy π_θ to the conditional denoising distribution $p_\theta(\cdot \mid x_t, q)$, and the trajectory τ to the sequence of intermediate noisy states $\{x_t\}$.

Specifically, for each prompt q, we sample a group of G roll-

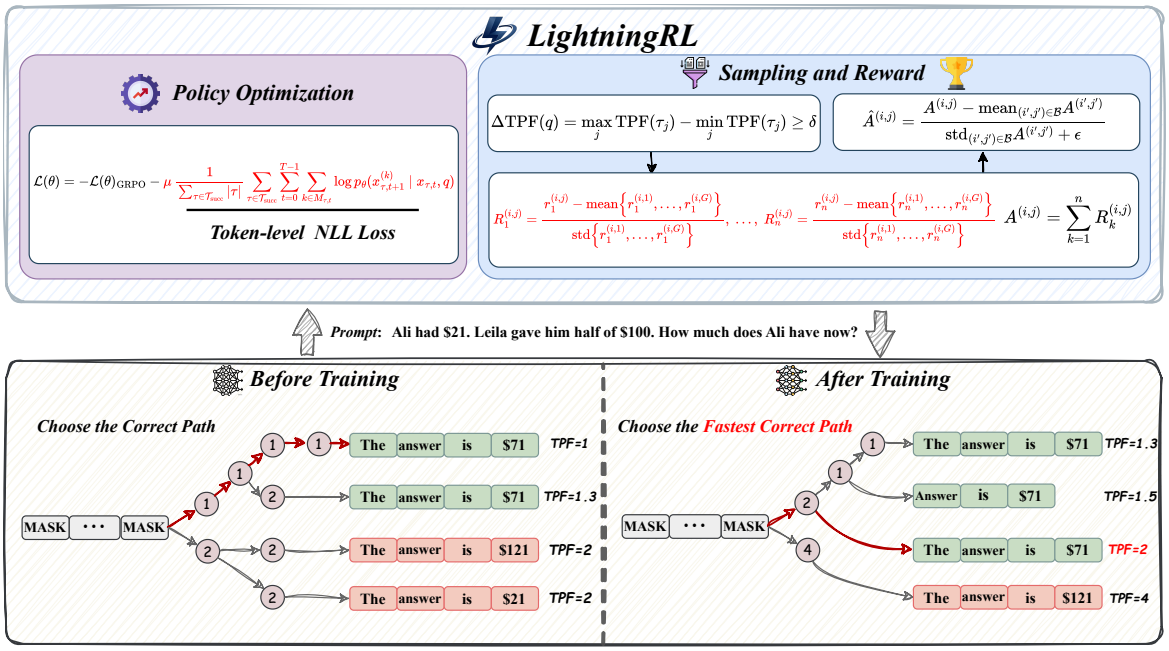


Figure 2. **Overview of LightningRL.** LightningRL samples a group of decoding trajectories per prompt, applies per-reward decoupled normalization to preserve within-group ranking under heterogeneous scales. The policy is optimized with a GRPO-style objective plus a token-level NLL anchor. The bottom panel shows the resulting shift toward the fastest correct trajectory, improving TPF without degrading accuracy.

outed trajectories $\{\tau_j\}_{j=1}^{G}$ using the behavior policy $\pi_{\theta_{\text{old}}}$. GRPO computes the terminal reward $R(\tau_j)$ and derives the advantage $\hat{A}_j$:

$$\hat{A}_j = \frac{R(\tau_j) - \frac{1}{G}\sum_{i=1}^{G} R(\tau_i)}{\text{std}(\{R(\tau_i)\}_{i=1}^{G}) + \epsilon}, \tag{2}$$

where ϵ is a small constant for numerical stability.

GRPO maximizes the advantage-weighted probability of valid actions. Substituting our dLLM-specific policy definition, the objective is formulated as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{j=1}^{G}\frac{1}{|\tau_j|}\sum_{t=0}^{T-1}\sum_{k\in M_{j,t}}\Bigg(\min\Big(\rho_{j,t,k}\hat{A}_j,\ \text{clip}(\rho_{j,t,k}, 1-\epsilon, 1+\epsilon)\hat{A}_j\Big) - \beta D_{\text{KL}}\big(p_\theta(\cdot|x_{j,t})\|p_{\text{ref}}(\cdot|x_{j,t})\big)\Bigg)\Bigg]. \tag{3}$$

Here, $M_{j,t} = \{i \mid x_{j,t}^{(i)} = \texttt{[MASK]}\}$ denotes the set of indices in the intermediate state $x_{j,t}$ that require denoising, consistent with the generation process defined in Eq. 1. Accordingly, $|\tau_j| = \sum_{t=0}^{T-1}|M_{j,t}|$ represents the total number of parallel token predictions performed along the trajectory. β is the coefficient for the KL divergence penalty against the reference model p_{ref}. The importance ratio $\rho_{j,t,k}$ is:

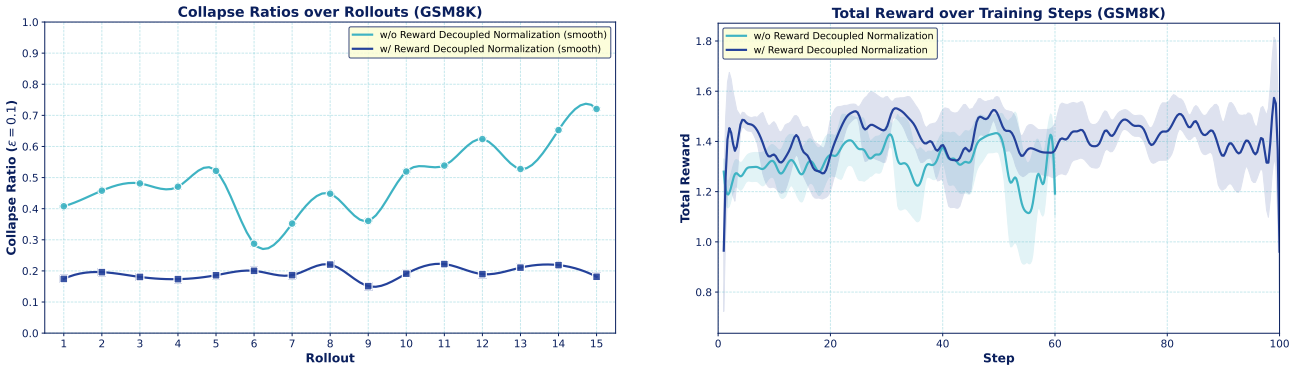
$$\rho_{j,t,k} = \frac{p_\theta(x_{j,t+1}^{(k)} \mid x_{j,t}, q)}{p_{\theta_{\text{old}}}(x_{j,t+1}^{(k)} \mid x_{j,t}, q)}. \tag{4}$$

The components marked in red highlight the structural differences from standard AR-LLMs — gradients are propagated through parallel masked positions over multiple denoising steps conditioned on the intermediate noisy states, rather than sequential token positions conditioned on history.

3. LightningRL: Breaking the Accuracy–Parallelism Trade-off

To push the frontier of dLLMs through RL, we initially adopted existing RL frameworks established for dLLMs (Zhao et al., 2025; Wang et al., 2025b; Zhu et al., 2026). However, we encountered significant challenges due to our multi-objective setting (i.e., simultaneously optimizing for accuracy and speed). We observed reward collapse, where one objective dominates the optimization. Thus, the policy drifts, and the model's generation capability degrades (see Sec. 4.4.1).

To address these, we present LightningRL, a robust training algorithm designed to co-optimize accuracy and inference parallelism. LightningRL introduces three specific modifications tailored for the dLLM landscape:



(a) Collapse ratio across rollouts on GSM8K (lower is better).

(b) Total reward trajectory during training on GSM8K.

Figure 3. **Per-reward decoupled normalization improves training stability.** It reduces signal collapse (a) and yields more stable reward optimization (b) under the same training setup.

- Per-reward Decoupled Normalization (Sec. 3.1), which mitigates reward collapse by independently normalizing distinct reward signals.
- Token-level NLL Regularization (Sec. 3.2), which is applied to correct trajectories to prevent policy drift and maintain linguistic coherence.
- Dynamic Sampling with TPF-aware Filtering (Sec. 3.3), which helps efficiently explore the trade-off between speed and quality.

3.1. Decoupled Normalization for Group Rewards

While GRPO works well for single-objective optimization, a naive multi-reward extension—summing rewards and then applying standard group-wise normalization—can fail. When a coarse, high-magnitude discrete reward (e.g., $\mathrm{Acc} \in \{-1, 1\}$) is mixed with a fine-grained continuous reward (e.g., the overall TPF of a sampling trajectory), the aggregate is often dominated by the discrete term, creating many within-group ties (or near-ties). This makes advantages indistinguishable across behaviors and weakens the effective policy-gradient signal.

To mitigate this issue, we propose per-reward decoupled normalization. Instead of normalizing the aggregated reward, we normalize each reward component independently within the group and then aggregate the standardized signals. Consider the i-th prompt in the current minibatch and its generated group of rollouts of size G. For the j-th rollout, let $r_k^{(i,j)}$ denote the raw value of the k-th reward objective among a total of n objectives. Instead of summing r_k, we first compute the independent advantage for each objective:

$$R_k^{(i,j)} = \frac{r_k^{(i,j)} - \mathrm{mean}\left\{r_k^{(i,1)}, \ldots, r_k^{(i,G)}\right\}}{\mathrm{std}\left\{r_k^{(i,1)}, \ldots, r_k^{(i,G)}\right\} + \epsilon}, \tag{5}$$

where ϵ is a small constant for numerical stability. This ensures that every reward component contributes an equally standardized ranking signal, regardless of its raw magnitude or distribution. The composite advantage is then derived by summing these normalized components:

$$A^{(i,j)} = \sum_{k=1}^{n} R_k^{(i,j)}. \tag{6}$$

Finally, we apply global batch-wise normalization to control the update scale regardless of the number of objectives, yielding the final advantage $\hat{A}^{(i,j)}$ used for policy updates:

$$\hat{A}^{(i,j)} = \frac{A^{(i,j)} - \mathrm{mean}_{(i',j')\in\mathcal{B}} A^{(i',j')}}{\mathrm{std}_{(i',j')\in\mathcal{B}} A^{(i',j')} + \epsilon}, \tag{7}$$

where $\mathcal{B}$ denotes the set of all rollout indices (i', j') in the current minibatch. Crucially, while the preceding decoupling step preserves group-relative distinctions contributed by each objective, this final normalization prevents the advantage magnitude from drifting with the reward dimensionality n.

To validate our motivation, we present two diagnostics. First, Fig. 3a shows that decoupled normalization substantially reduces the Collapse Ratio, defined as the proportion of within-group pairs with advantage differences below ϵ, thereby preserving finer preference resolution. Second, as shown in Fig. 3b, removing this decoupling step leads to noisier optimization trajectories, slower convergence, and lower peak performance. These results confirm that decoupling improves the conditioning of aggregated advantages by maintaining meaningful distinctions while stabilizing the update scale.

3.2. Token-Level NLL for Accuracy Anchoring

In multi-objective RL with verifier-based supervision, rewards on math/code tasks are typically sparse and sequence-

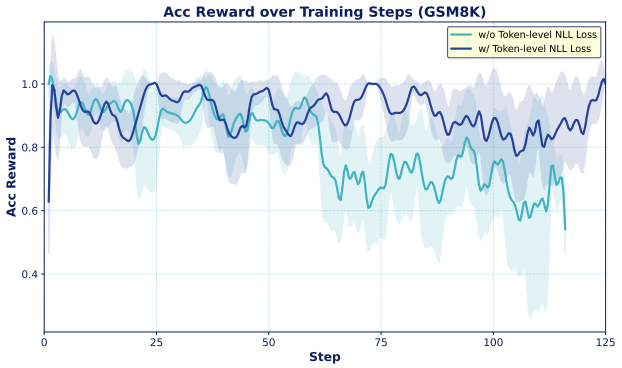


Figure 4. **Token-level NLL loss anchors the accuracy objective on GSM8K.** Compared with training without the token-level NLL term, it maintains a higher accuracy reward and mitigates late-stage drift in the accuracy signal under the same setup.

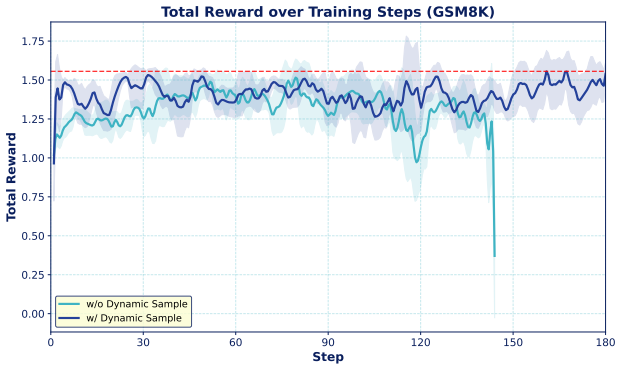


Figure 5. **Dynamic sampling improves robustness of total-reward optimization on GSM8K.** Relative to training without dynamic sampling, it reduces reward collapse under the same setup.

level. In this setting, the policy gradient estimator necessarily takes a score-function form, where a scalar advantage multiplies log-probability gradients of the actions taken along the trajectory.

Formally, GRPO assigns each sampled trajectory τ a single group-relative advantage $\hat{A}(\tau)$. Abstracting away the piecewise clipping details, the induced update admits the canonical score-function structure:

$$\nabla_\theta \mathcal{L}_{\text{GRPO}} \propto -\mathbb{E}\left[\frac{\hat{A}(\tau)}{|\tau|}\sum_{t=0}^{T-1}\sum_{k\in M_t}\nabla_\theta \log p_\theta(x_{t+1}^{(k)} \mid x_t, q)\right] \tag{8}$$

Crucially, $\hat{A}(\tau)$ is shared by all decoding steps in a trajectory: the factor $1/|\tau|$ only rescales the update and does not refine credit assignment. In multi-objective training, the *accuracy-driven* gradient can be substantially attenuated by per-token dilution and further overshadowed when non-accuracy rewards dominate the effective advantage. This weakens the corrective pressure toward correctness and can induce drift into fast-but-incorrect modes. This motivates a positive-example LM loss that turns verifier-correct trajectories into dense, token-factorized anchoring toward correct behaviors.

To maximize the utility of rare verified successes and explicitly anchor the policy toward correctness, we introduce a token-level NLL loss that converts sequence-level successes into dense token-factorized supervision. Let $\mathcal{T}_{\text{succ}}$ denote the set of verifier-correct trajectories among on-policy samples in the current update. We define:

$$\mathcal{L}_{\text{NLL}}(\theta) = -\frac{1}{\sum_{\tau\in\mathcal{T}_{\text{succ}}}|\tau|}\sum_{\tau\in\mathcal{T}_{\text{succ}}} \sum_{t=0}^{T-1}\sum_{k\in M_{\tau,t}}\log p_\theta(x_{\tau,t+1}^{(k)} \mid x_{\tau,t}, q), \tag{9}$$

and set $\mathcal{L}_{\text{NLL}}(\theta) = 0$ when $\mathcal{T}_{\text{succ}} = \emptyset$. Here, consistent with Sec. 2.1, $M_{\tau,t}$ denotes the masked indices at step t for trajectory τ. Overall, we optimize the combined objective:

$$\mathcal{L}_{\text{LightningRL}}(\theta) = \mathcal{L}_{\text{GRPO}}(\theta) + \mu\,\mathcal{L}_{\text{NLL}}(\theta), \tag{10}$$

where μ is a scalar coefficient balancing the two losses. This yields a clean division of labor: $\nabla\mathcal{L}_{\text{GRPO}}$ drives preference learning via relative ranking across sampled trajectories, while $\mu\nabla\mathcal{L}_{\text{NLL}}$ acts as a self-imitation anchor that reallocates probability mass onto verifier-correct trajectories with stable, token-factorized gradients. Fig. 4 supports this anchoring effect: while training without token-level NLL loss exhibits a pronounced downward drift, the anchored variant maintains consistently higher accuracy rewards and substantially improved stability over training steps.

3.3. Dynamic Sampling for Efficient Policy Optimization

Standard GRPO can suffer from gradient starvation under reward quantization. GRPO estimates advantages based on within-group relative rewards; consequently, when the G sampled trajectories $\{\tau_j\}_{j=1}^{G}$ for a prompt q fall into the same reward bin, the relative advantages vanish. This issue is particularly pronounced in our multi-objective setting: once the accuracy reward saturates within a group, differentiation relies solely on the speed reward, which is derived from discrete TPF (Token Passing Fraction) values. As a result, many groups yield identical total rewards $R(\tau_j)$, leading to vanishing gradients and wasted rollout budgets.

To mitigate this issue, we propose dynamic sampling with TPF-aware filtering. We define the within-group TPF spread as:

$$\Delta\text{TPF}(q) = \max_j \text{TPF}(\tau_j) - \min_j \text{TPF}(\tau_j) \geq \delta, \tag{11}$$

where δ is a predefined filtering threshold. For each candidate prompt, we first sample a group of trajectories and

Table 1. **Evaluation results of LightningRL and the baselines on math and code benchmarks.** The notation *LightningRL-8B-b32* denotes the 8B LightningRL model with a block size of 32. LightningRL consistently advances the speed–quality frontier over its SDAR baseline and prior diffusion and autoregressive baselines, achieving substantially higher AUP at comparable accuracy.

Model	GSM8K			MATH500			MBPP			HumanEval		
	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP
dLLMs												
Dream (Ye et al., 2025)	83.9	1.00	83.9	39.6	1.00	39.6	57.2	1.00	57.2	55.2	1.00	55.2
Fast-dLLM-Dream (Wu et al., 2025b)	79.0	1.44	116.5	38.3	1.78	55.2	53.2	1.20	63.6	54.3	1.33	63.5
dParalel-Dream (Chen et al., 2025)	82.1	3.02	245.7	38.7	2.94	77.9	55.4	2.24	108.0	54.3	2.57	98.8
d3LLM-Dream (Qian et al., 2026)	81.4	4.94	391.3	38.2	3.92	97.5	55.6	2.96	141.4	57.1	3.20	129.5
LLaDA (Nie et al., 2025)	72.6	1.00	72.6	32.2	1.00	32.2	47.1	1.00	47.1	38.3	1.00	38.3
Fast-dLLM-LLaDA (Wu et al., 2025b)	74.7	2.77	205.8	30.8	1.97	47.2	38.6	2.13	56.6	37.8	2.56	54.0
D2F-LLaDA (Wang et al., 2025a)	73.2	2.88	209.7	28.7	2.38	45.5	38.0	1.94	50.0	36.6	2.69	62.0
dParallel-LLaDA (Chen et al., 2025)	72.6	5.14	358.1	30.2	3.17	64.5	40.0	2.35	60.5	39.0	4.93	83.7
AR Models												
Qwen-2.5-7B-it (Qwen et al., 2025)	74.1	1.00	74.1	41.4	1.00	41.1	**63.6**	1.00	63.6	67.7	1.00	67.7
EAGLE-3 (LLaMA-3.1) (Li et al., 2025)	76.6	5.12	319.0	39.8	5.72	142.1	60.2	5.69	298.6	67.6	5.98	344.8
Block-wise dLLMs												
Fast-dLLM-v2 (Wu et al., 2025a)	77.5	2.21	156.0	48.7	2.61	126.7	50.1	2.04	81.9	61.7	2.58	128.9
SDAR-8B-b32 (Cheng et al., 2025)	88.9	2.85	252.5	**63.6**	4.81	299.5	58.0	2.44	81.1	**73.5**	2.39	123.8
LightningRL-8B-b32	**90.3**	**5.58**	**492.4**	63.0	**6.28**	**407.5**	58.3	**11.10**	**641.6**	72.6	**6.30**	**450.1**

accept the prompt only if it satisfies the above criterion; otherwise, we discard it and resample a new prompt. We repeat this accept–reject process until the batch is filled. By filtering out near-tie groups, this procedure keeps a more consistent fraction of non-zero advantages in each update, leading to denser and more stable policy gradient signals.

Empirically, Fig. 5 shows that without dynamic sampling the training curve converges more slowly, reaches a lower plateau, and eventually undergoes a pronounced collapse, whereas with dynamic sampling training remains stable and achieves faster, higher-reward convergence under the same configuration.

4. Experiments

4.1. Setup

Model and Dataset We implement LightningRL upon the SDAR model family (Cheng et al., 2025) due to its leading performance. We refer to the 8B SDAR model with a block size of 32 as SDAR-8B-b32, with all other variants named likewise. For training, we utilize the training split of the MATH (Hendrycks et al., 2021) dataset for mathematical reasoning and the PrimeIntellect (Team et al., 2025) dataset for code generation tasks.

Reinforcement Learning Settings During data collection, we use a batch size of 128 tasks and a group size of 32 responses per task. We employ a low confidence dynamic sampling strategy with a threshold $\phi = 0.9$ and a sampling temperature of 1.0.

Benchmark and Baselines We evaluate on four representative benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021). To ensure a fair comparison with prior work, we employ a 4-shot setting for MATH and a 3-shot setting for Llada-based models (Nie et al., 2025) on MBPP. All other evaluations are conducted in a zero-shot setting. We benchmark our approach against state-of-the-art dLLMs and AR models.

Evaluation Metrics We assess performance using three key metrics: TPF (for Parallelism), Accuracy, and the AUP (Qian et al., 2026) score.

4.2. Main Results

An illustration of the comparison between LightningRL and leading baselines is provided in Fig. , with a complete comparison with more baselines detailed in Tab. 1. As shown, LightningRL-8B-b32 achieves an average AUP of 497.9 and an average TPF of 7.32 on these baselines, with an average accuracy of 71.1%. In comparison, the SDAR-8B-b32 baseline attains only an average AUP of 189.2 and a TPF of 3.12 while remaining almost identical average accuracy (71.0%). Moreover, LightningRL consistently dominates representative AR and diffusion baselines, surpassing EAGLE-3 (Li et al., 2025) (276.1 average AUP, 5.63 average TPF) and the d3LLM (Qian et al., 2026) family across the same evaluation settings.

4.3. Scalability and Efficiency Analysis

As summarized in Tab. 2, we evaluate scalability along two primary axes. First, regarding model scale: under a fixed block size of 32, scaling from 1.7B to 8B parameters significantly boosts both reasoning accuracy and parallelism. Notably, while the baseline's throughput remains stagnant as model size grows, LightningRL effectively leverages

Table 2. **Comparison of SDAR and LightningRL under identical settings.** The results are grouped by Model Scale and Block Size (BS) to facilitate direct comparison across four datasets.

Scale	BS	Model	GSM8K			MATH500			MBPP			HumanEval		
			Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP
1.7B	32	SDAR	71.5	2.48	176.4	41.2	5.62	226.3	39.0	3.86	149.8	48.8	2.81	136.0
		LightningRL	71.7	**3.40**	**251.4**	41.2	**6.01**	**241.2**	37.3	**5.07**	**185.8**	48.2	**3.43**	**165.2**
4B	32	SDAR	86.6	3.10	243.8	53.6	5.09	263.9	54.0	1.96	105.9	59.8	3.67	216.8
		LightningRL	85.4	**4.37**	**374.7**	56.4	**6.55**	**358.7**	52.2	**3.05**	**158.3**	57.3	**4.35**	**242.3**
8B	32	SDAR	88.9	2.85	252.5	63.6	4.81	299.5	58.0	2.44	81.1	74.4	2.39	123.8
		LightningRL	90.3	**5.58**	**492.4**	63.0	**6.28**	**407.5**	58.3	**11.10**	**641.6**	72.6	**6.30**	**450.1**
	8	SDAR	91.0	2.96	269.1	64.1	3.53	224.8	59.6	1.72	103.3	76.9	2.77	211.6
		LightningRL	89.4	**3.75**	**331.8**	67.0	**4.21**	**279.5**	58.2	**3.11**	**179.3**	76.8	**3.30**	**258.8**
	4	SDAR	91.1	2.35	213.9	71.2	2.49	176.8	63.3	1.84	116.7	78.6	1.53	120.3
		LightningRL	91.0	**3.21**	**291.6**	70.3	**3.42**	**237.7**	63.2	**2.47**	**155.5**	78.2	**2.32**	**181.3**

increased capacity to unlock higher parallelism. Second, regarding block size: increasing the block size on the 8B model raises the upper bound for parallel decoding, an effect especially prominent on the MBPP benchmark. Overall, LightningRL scales favorably and remains robustly effective across different model sizes and decoding configurations.

4.4. Qualitative Analysis

4.4.1. Training Dynamics

We plot the training curves of LightningRL on GSM8K in Fig. 6a-6c. For comparison, we apply TraceRL (Wang et al., 2025b), a commonly used RL framework for dLLMs, to the same training settings (including rewards, hyperparameters, etc.). As shown, during the training of TraceRL, the optimization for speed progressively erodes the correctness signal, and the accuracy reward drops sharply while speed gains remain limited. The training ultimately collapses. In contrast, LightningRL converges stably and avoids this collapse behavior, maintaining the accuracy signal while improving speed, resulting in a consistently better efficiency–accuracy frontier.

4.4.2. Analysis of Decoding Behavior

Fig. 6d analyzes the step-wise decoding dynamics of the model trained by LightningRL. Compared to the SDAR baseline, LightningRL finishes decoding of various samples much more synchronously, with most samples terminating within $\sim$100 steps, whereas SDAR exhibits a heavy tail extending beyond 200 steps (dashed lines). The gain comes from higher throughput during the active phase: in the first 0–100 steps, LightningRL sustains roughly 2× TPF than SDAR (5 vs. 2.5), and it later ramps to 20 TPF to quickly consume the remaining positions. As a result, LightningRL avoids the quasi-serial slowdown of SDAR and largely removes long-tail decoding steps.

Table 3. **Ablation study of LightningRL components on GSM8K.** We report the impact of each proposed module on reasoning accuracy (Acc) and inference parallelism (TPF). The results show that each component is critical to maintaining the optimal Pareto frontier.

Case	Acc (%)	TPF
w/o NLL loss	84.7	4.43
w/o per-reward decoupled normalization	83.2	4.79
w/o TPF-aware filtering	84.1	4.96
LightningRL (Full)	**90.3**	**5.58**

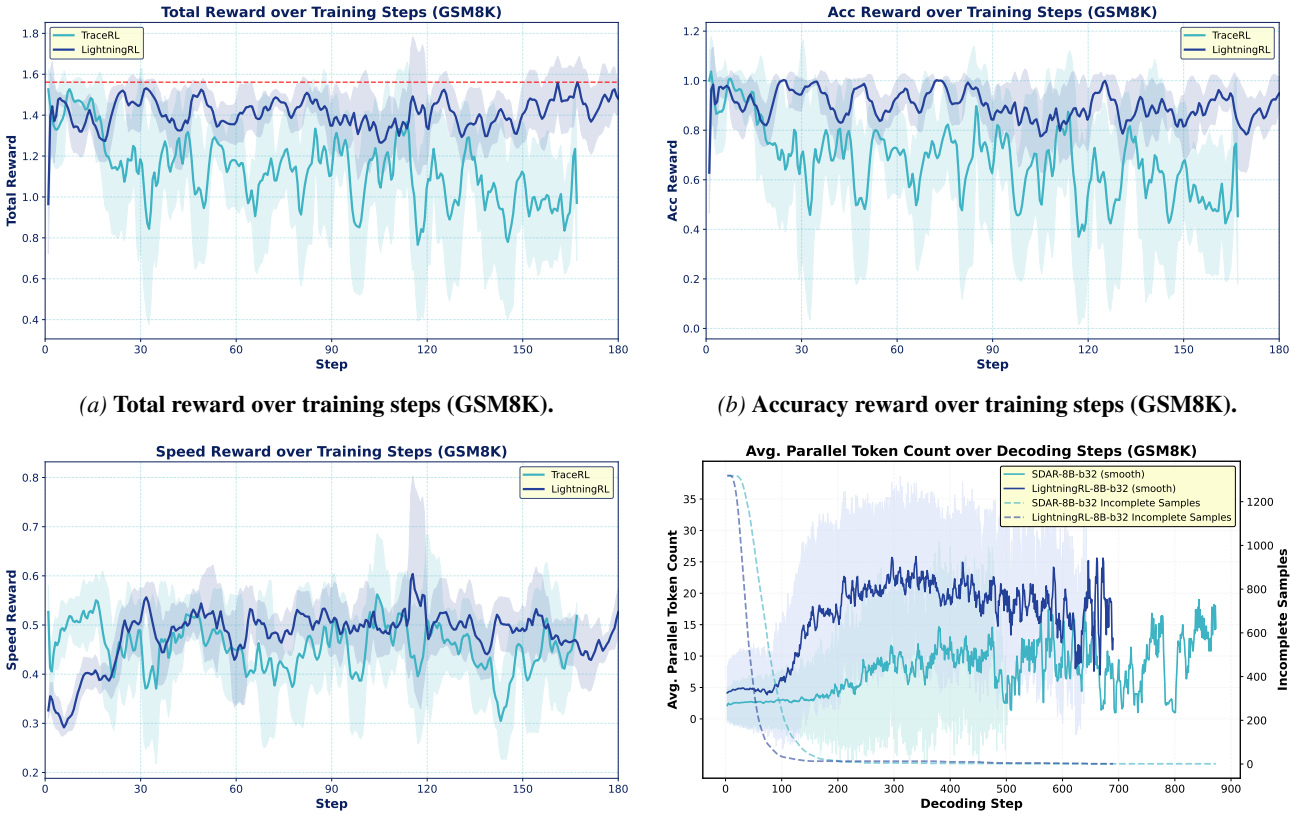
4.5. Ablation

Tab. 3 reports ablations of the three key components of LightningRL on GSM8K using the 8B model with block size 32. LightningRL achieves the best operating point, reaching 90.3% accuracy at 5.58 TPF. Removing the NLL loss reduces both accuracy and parallelism (84.7%, 4.43 TPF), indicating that the NLL term helps anchor correctness under aggressive parallel decoding. Without per-reward decoupled normalization, performance drops to 83.2% and 4.79 TPF, suggesting that properly scaling multi-objective advantages is important for stable optimization. Disabling TPF-aware filtering also degrades results (84.1%, 4.96 TPF), implying that filtering low-contrast groups improves the effectiveness of the training signal.

5. Related Work

5.1. dLLM Acceleration

dLLMs offer a non-autoregressive generation paradigm that iteratively denoises a fully or partially masked sequence, enabling parallel token updates but leaving practical inference efficiency still comparatively underexplored. Recent acceleration work largely falls into: (i) reducing per-step compute via diffusion-compatible caching and confidence-aware decoding rules (Wu et al., 2025b;a); (ii) increasing



(a) **Total reward over training steps (GSM8K).**

(b) **Accuracy reward over training steps (GSM8K).**

(c) **Speed reward over training steps (GSM8K).**

(d) **Decoding behavior.** LightningRL maintains higher throughput and reduces long-tail decoding steps relative to the baseline.

Figure 6. **Training dynamics and decoding behavior on GSM8K.** LightningRL avoids the objective drift and reward collapse observed in TraceRL, while sustaining higher decoding throughput and more synchronized termination.

effective parallelism and/or cutting denoising steps through improved unmasking, sampling, and learned parallel decoding strategies (Xu et al., 2025; Bao et al., 2025; Chen et al., 2025); and (iii) hybrid diffusion–autoregressive pipelines that better leverage block-wise decoding and KV-cache-style components (Wang et al., 2025a). Our work builds upon these acceleration techniques but focuses on optimizing the decoding trajectory via reinforcement learning to mitigate the accuracy loss typically associated with high parallelism.

5.2. Reinforcement Learning in dLLMs

Recent advances in Diffusion Large Language Models (dLLMs) leverage RL for step-wise denoising. Beyond early masked SFT and policy optimization (Zhao et al., 2025; Gong et al., 2025), research has shifted toward trajectory-level optimization, including trace-aware RL (Wang et al., 2025b), step-aligned scheduling (He et al., 2025), and outcome-driven reasoning (Huang et al., 2025). Recent refinements further incorporate accuracy-aware signals (Ma et al., 2024) or joint control-variable optimization (Zhou et al., 2025). Despite these gains, existing frameworks prioritize generation quality and relegate parallelism to a mere inference-time adjustment. LightningRL addresses this gap by treating parallelism as a first-class training objective, employing multi-objective RL to jointly optimize speed and accuracy. Decoupled and anchored regularization adopted by LightningRL stabilizes multi-objective training against sparse-reward instability.

6. Conclusion

In this paper, we presented LightningRL, an RL framework designed to optimize the speed–quality trade-off in diffusion language models. By mitigating the error amplification inherent in aggressive parallel decoding, LightningRL reconciles the conflict between generation speed and accuracy. Our results on maths and code generation benchmarks demonstrate that LightningRL consistently achieves higher AUP under high parallelism constraints, paving the way for practical, high-throughput dLLM deployment. Future work will explore scaling laws with larger contexts and generalize this approach to a wider range of dLLM generation tasks.

Impact Statement

This paper presents foundational research in deep learning, aiming to advance the field of machine learning. While optimizing the accuracy–parallelism frontier and inference efficiency of Diffusion Large Language Models via reinforcement learning may yield potential societal impacts, we do not engage in a detailed discussion here, as the ethical implications and anticipated societal consequences of developments in efficient generative AI are already well-established.

References

Arriola, M., Gokaslan, A., Chiu, J. T., Yang, Z., Qi, Z., Han, J., Sahoo, S. S., and Kuleshov, V. Block diffusion: Interpolating between autoregressive and diffusion language models, 2025. URL https://arxiv.org/abs/2503.09573.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Bao, W., Chen, Z., Xu, D., and Shang, Y. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding, 2025. URL https://arxiv.org/abs/2509.25188.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. 2021.

Chen, Z., Fang, G., Ma, X., Yu, R., and Wang, X. dparallel: Learnable parallel decoding for dllms. *arXiv preprint arXiv:2509.26488*, 2025.

Cheng, S., Bian, Y., Liu, D., Zhang, L., Yao, Q., Tian, Z., Wang, W., Guo, Q., Chen, K., Qi, B., and Zhou, B. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. URL https://arxiv.org/abs/2510.06303.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gong, S., Li, M., Feng, J., Wu, Z., and Kong, L. Diffuseq: Sequence to sequence text generation with diffusion models, 2023. URL https://arxiv.org/abs/2210.08933.

Gong, S., Zhang, R., Zheng, H., Gu, J., Jaitly, N., Kong, L., and Zhang, Y. Diffucoder: Understanding and improving masked diffusion models for code generation, 2025. URL https://arxiv.org/abs/2506.20639.

Gulrajani, I. and Hashimoto, T. B. Likelihood-based diffusion language models, 2023. URL https://arxiv.org/abs/2305.18619.

He, H., Renz, K., Cao, Y., and Geiger, A. Mdpo: Overcoming the training-inference divide of masked diffusion language models, 2025. URL https://arxiv.org/abs/2508.13148.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huang, Z., Chen, Z., Wang, Z., Li, T., and Qi, G.-J. Reinforcing the diffusion chain of lateral thought with diffusion language models, 2025. URL https://arxiv.org/abs/2505.10446.

Kou, S., Hu, L., He, Z., Deng, Z., and Zhang, H. Cllms: Consistency large language models. In *Forty-first International Conference on Machine Learning*, 2024.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. B. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL https://arxiv.org/abs/2503.01840.

Ma, Y., Melnychuk, V., Schweisthal, J., and Feuerriegel, S. Diffpo: A causal diffusion model for learning distributions of potential outcomes, 2024. URL https://arxiv.org/abs/2410.08924.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Qian, Y.-Y., Su, J., Hu, L., Zhang, P., Deng, Z., Zhao, P., and Zhang, H. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation, 2026. URL https://arxiv.org/abs/2601.07568.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models, 2024a. URL https://arxiv.org/abs/2406.07524.

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models, 2024b. URL https://arxiv.org/abs/2406.07524.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Team, P. I., Senghaas, M., Obeid, F., Jaghouar, S., Brown, W., Ong, J. M., Auras, D., Sirovatka, M., Straube, J., Baker, A., Müller, S., Mattern, J., Basra, M., Ismail, A., Scherm, D., Miller, C., Patel, A., Kirsten, S., Sieg, M., Reetz, C., Erdem, K., Weisser, V., and Hagemann, J. Intellect-3: Technical report, 2025. URL https://arxiv.org/abs/2512.16144.

Wang, X., Xu, C., Jin, Y., Jin, J., Zhang, H., and Deng, Z. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing, 2025a. URL https://arxiv.org/abs/2508.09192.

Wang, Y., Yang, L., Li, B., Tian, Y., Shen, K., and Wang, M. Revolutionizing reinforcement learning framework for diffusion large language models, 2025b. URL https://arxiv.org/abs/2509.06949.

Wu, C., Zhang, H., Xue, S., Diao, S., Fu, Y., Liu, Z., Molchanov, P., Luo, P., Han, S., and Xie, E. Fast-dllm v2: Efficient block-diffusion llm, 2025a. URL https://arxiv.org/abs/2509.26328.

Wu, C., Zhang, H., Xue, S., Liu, Z., Diao, S., Zhu, L., Luo, P., Han, S., and Xie, E. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025b. URL https://arxiv.org/abs/2505.22618.

Xu, C., Jin, Y., Li, J., Tu, Y., Long, G., Tu, D., Song, M., Si, H., Hou, T., Yan, J., and Deng, Z. Lopa: Scaling dllm inference via lookahead parallel decoding, 2025. URL https://arxiv.org/abs/2512.16229.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Zhao, S., Gupta, D., Zheng, Q., and Grover, A. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.12216.

Zhou, R., Ni, Z., Chen, T., Liu, Z., Yue, Y., Wang, Y., Wang, Y., Liu, J., and Huang, G. Co-grpo: Co-optimized group relative policy optimization for masked diffusion model, 2025. URL https://arxiv.org/abs/2512.22288.

Zhu, Y., Wan, J., Liu, X., He, S., Wang, Q., Guo, X., Liang, T., Huang, Z., He, Z., and Qiu, X. Dirl: An efficient post-training framework for diffusion language models, 2026. URL https://arxiv.org/abs/2512.22234.

A. Discussion on Value Model Incorporation

To further explore the potential of traditional reinforcement learning components within our framework, we investigated the integration of a dedicated value model for advantage estimation. As shown in Tab. 4, we compared the performance of a configuration incorporating a value model against the full LightningRL framework.

Method	**Acc** (%)	**TPF**
w/ Value Model	35.1	4.40
LightningRL	**90.3**	**5.58**

Table 4. Comparison of Performance with and without Value Model Incorporation on GSM8K.

Our experimental results indicate that the inclusion of a value model is currently ineffective for this architecture, leading to a substantial decline in both reasoning accuracy and generation efficiency (TPF). We hypothesize that this performance drop stems from the inherent difficulty in accurately estimating value functions within complex, multi-step reasoning trajectories. Inaccurate value estimations likely introduce significant bias during the policy update process, thereby hindering the model's convergence. Further investigation is required to develop more robust value estimation techniques suitable for high-throughput reasoning tasks.