

Rethinking Diffusion Large Language Models via Time-aware Weighting and Block-oriented Refinements

Zheyong Xie¹, Shengrui Li¹, Li Zhong¹, Kaiyan Zhao^{1,2}, Fangcheng Shi¹, Jiakang Wang¹, Huayi Jin¹, Zhaokai Luo¹, Fei Zhao¹, Haifeng Liu¹, Jieying Ye¹, Chonggang Lu¹, Wenxuan Huang¹, Yue Liu¹, Zijie Meng¹, Shaosheng Cao^{1,3*}

¹Xiaohongshu Inc., ²The University of Tokyo, ³Tsinghua University

caoshaosheng@xiaohongshu.com

Abstract

Diffusion Large Language Models (DLLMs) have recently emerged as a parallel alternative to autoregressive models, offering bidirectional context modeling and parallel decoding. However, developing highly performant DLLMs is hindered by lifecycle challenges, including suboptimal timestep weighting in pre-training, train-inference distribution gaps in post-training, and underutilized caching during inference. To address these obstacles, we introduce ReDLLM, a 30B-parameter Mixture-of-Experts DLLM. It incorporates a time-aware weighting strategy during continual pre-training that scales gradients according to denoising difficulty. Furthermore, we propose two block-oriented refinements: a terminal-block treatment to narrow the train-inference gap, and a block-aware inference pipeline featuring inline block prefill and block-level scheduling to maximize KV-cache reuse and parallelization. Extensive evaluations demonstrate that ReDLLM surpasses existing DLLMs at the same active-parameter scale, achieves generation quality competitive with 100B DLLMs, and attains the highest inference throughput among evaluated models.

1 Introduction

Large-scale autoregressive (AR) language models have become the dominant paradigm in Natural Language Processing, achieving strong performance across a wide range of tasks (Grattafiori et al., 2024; Hurst et al., 2024; Singh et al., 2025; Guo et al., 2025a; Zhao et al., 2025a,b). However, their strict left-to-right decoding limits inference parallelism and bidirectional context modeling (Tay et al., 2022). As a parallel alternative, Diffusion Large Language Models (DLLMs) have recently emerged as a promising direction (Ye et al., 2025; Nie et al., 2025b): by formulating text generation as iterative denoising over a masked sequence, they

enable bidirectional context modeling and parallel decoding, and have been shown to rival AR models in quality while offering additional flexibility on tasks such as text infilling and constrained generation (Nie et al., 2025b; Bie et al., 2025; Liu et al., 2025; Bie et al., 2026; Cheng et al., 2025a,b).

However, building a strong DLLM across pre-training, post-training, and inference remains difficult, and we identify three obstacles that are not adequately addressed by prior work. **First**, in the pre-training of DLLMs, the standard masked diffusion objective (Nie et al., 2025b) applies uniform weights across all timesteps, which is suboptimal because timesteps with less noise (masks) are easier for models to learn than highly masked ones. **Second**, DLLMs face a distribution gap when handling the last partially filled block of a sequence. Specifically, the empty positions contain padding tokens during training, but are filled with mask during inference. **Third**, while block-wise masked diffusion (Arriola et al., 2025) enables parallel decoding and KV-cache reuse, current inference pipelines do not take advantage of the boundaries between blocks for optimal caching and request scheduling.

We address these obstacles in a single model, ReDLLM, with three corresponding solutions. **(i) Time-aware weighting for continual pre-training.** We augment the masked diffusion loss with a re-weighting term that increases the contribution of low-noise timesteps. By doing so, we explicitly encourage the model to learn from easier, near-clean data, while still receiving signal from highly corrupted cases. **(ii) Block-oriented post-training.** We first narrow the train-inference gap in post-training by filling incomplete terminal blocks with [EOS] tokens before applying random masking during both SFT and RL. We then apply an adapted trajectory-aware RL stage (Wang et al., 2025), which brings substantial gains on mathematical and reasoning tasks. **(iii) Block-aware inference.** The proposed inference procedure ex-

*Project lead and corresponding author.

exploits the block-wise structure of DLLM generation through two key optimizations. First, the KV caching of a generated block is directly fused with the denoising step of the subsequent block, avoiding the need for an extra forward pass. Second, request scheduling is shifted to block level, preventing faster requests from being bottlenecked by slower ones within the same batch. Together, these designs substantially reduce inference latency without affecting generation quality.

Instantiated as a 30B-parameter MoE model with 3B active parameters, ReDLLM is trained end-to-end with the above pipeline. Across extensive benchmarks, ReDLLM surpasses DLLMs at the same active-parameter scale, achieves competitive quality with much larger 100B-scale DLLMs, and attains the highest inference throughput among all evaluated models. Moreover, we provide controlled ablations isolating the contribution of each of the three components: time-aware weighting improves average benchmark performance, the [EOS]-based terminal-block treatment outperforms [MASK] filling by 5.96 points, and the two inference designs together deliver a $2.65\times$ throughput improvement at batch size 128.

In summary, our contributions are as follows:

- We introduce a time-aware weighting factor that scales gradients according to the denoising difficulty during continual pre-training from an AR checkpoint.
- We propose two block-oriented refinements: a terminal-block treatment to mitigate the train-inference gap for SFT and RL; and a block-wise generation pipeline via inline block prefill and block-level scheduling, substantially improving throughput without sacrificing quality.
- ReDLLM surpasses existing DLLMs at the same active-parameter scale, remains competitive with much larger scale DLLMs, and attains the highest inference throughput among all evaluated models.

2 Related Work

Early research on text diffusion focused on adapting diffusion processes to discrete state spaces. D3PM (Austin et al., 2021a) generalized the diffusion framework to discrete spaces via transition matrices, while RADD (Ou et al., 2026) introduced absorbing states and time-independent conditional

models, establishing the loss formulation widely adopted in subsequent work. Shi et al. (2024) further extended these formulations to continuous-time frameworks, and SMDM (Nie et al., 2025a) empirically demonstrated that masked diffusion models follow scaling laws comparable to autoregressive models.

Building on these foundations, recent efforts have scaled DLLMs to billions of parameters, including LLaDA (Nie et al., 2025b; Bie et al., 2025, 2026), Dream (Ye et al., 2025), SDAR (Cheng et al., 2025a), and WeDLM (Liu et al., 2025). In parallel, a growing body of work has explored diverse technical directions for DLLMs, ranging from unified modeling formulations (Liu et al., 2026) and special-token-based mechanisms for improving generation quality (Zhang et al., 2026) to adaptive length expansion (Li et al., 2026a). In contrast to these works that primarily focus on modeling, ReDLLM provides a complete pre-training, post-training, and inference solution, addressing robustness, efficiency, and deployment challenges.

3 ReDLLM

This section introduces the overall pipeline of ReDLLM. We begin with the problem definition and theoretical basis of DLLMs, then detail the four stages as shown in Figure 1: Continual Pre-training, SFT, RL, and Inference, with each stage building upon the previous one.

3.1 Preliminary

Autoregressive models factorize the joint probability distribution of a text sequence $\mathbf{x} = [x^1, \dots, x^L]$ as a product of conditional probabilities over preceding tokens: $P(\mathbf{x}) = \prod_{i=1}^L P(x^i \mid x^{<i}; \theta)$, where θ stands for the model parameters. In contrast, DLLMs model the joint distribution via a *forward* corruption process and a *reverse* reconstruction process. In the forward process, each token is independently replaced by a [MASK] token with probability $t \in (0, 1]$, with t controlling the corruption level. The reverse process learns to recover the masked tokens as t decreases from 1 to 0. Most DLLMs (Li et al., 2025) parameterize a generative model p_θ to fit this reverse transition, predicting the original tokens given the corrupted sequence x_t and time t . Following this paradigm, prior work (Nie et al., 2025a) formulates the training objective as:

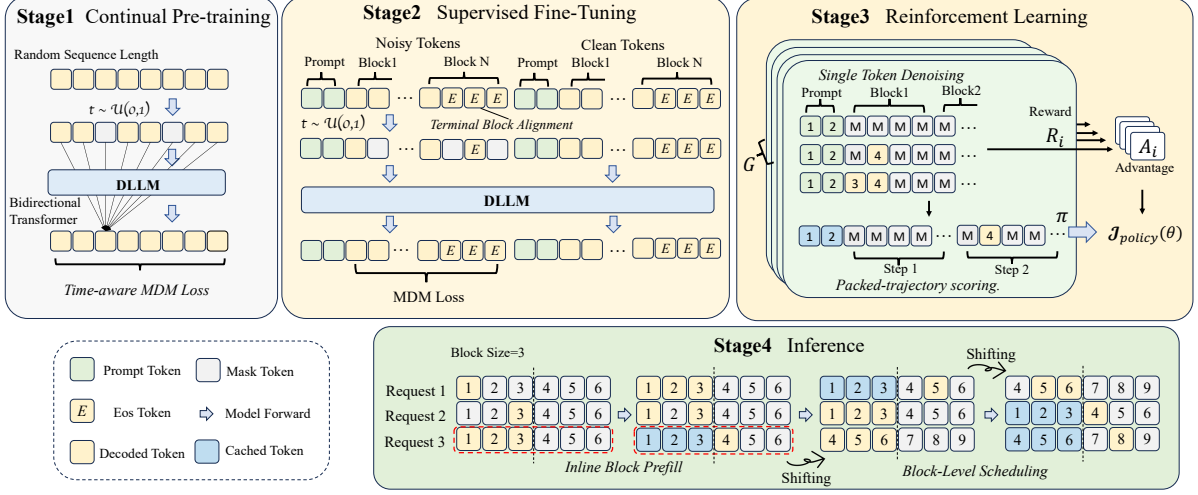


Figure 1: Overview of the ReDLLM training and inference paradigm. Stage 1 performs continual pre-training on random-length sequences using a time-aware MDM objective. Stage 2 conducts supervised fine-tuning with blockwise denoising and terminal block alignment. Stage 3 applies reinforcement learning on single-token denoising, optimizing the policy with packed-trajectory scoring. Stage 4 details the inference process, employing inline block prefill and block-level scheduling to accelerate generation efficiency.

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t} \left[\frac{1}{t} \sum_{i=1}^L \mathbf{1}[x_t^i = M] \log p_\theta(x_0^i | x_t) \right], \quad (1)$$

where $\mathbf{1}[\cdot]$ restricts the loss to masked positions and the L is the sequence length. This objective serves as an upper bound on the negative log-likelihood:

$$-\mathbb{E}_{p_{\text{data}}(x_0)} [\log p_\theta(x_0)] \leq \mathcal{L}(\theta), \quad (2)$$

so that minimizing $\mathcal{L}(\theta)$ implicitly maximizes the data likelihood. Building on this foundation, we introduce **ReDLLM**, which achieves strong generation quality and high decoding throughput through robust engineering and optimized training.

3.2 Time-aware Weighting Pre-training

A central challenge when continually pre-training DLLM from a pre-trained AR checkpoint is that the standard MDM (Masked Diffusion Models) objective in Equation 1 treats all noise levels as equally important, whereas the actual difficulty of recovering x_0 from x_t varies sharply with t . At low t , x_t retains most of the original context and recovering the few masked tokens is much easier than high t samples, which are dominated by [MASK]. We therefore introduce a time-aware weighting factor $\frac{1}{t}$ to encourage the model to receive stronger supervision from easier cases, while still receiving signal from harder ones, yielding:

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t} \left[\frac{1}{t} \cdot \left(\frac{1}{t} \sum_{i=1}^L \mathbf{1}[x_t^i = M] \log p_\theta(x_0^i | x_t) \right) \right]. \quad (3)$$

Here, the factor $\frac{1}{t}$ inside is inherited from the standard MDM objective, while the outside factor is the additional time-aware weighting introduced in our method. This static re-weighting increases the relative gradient contribution of low-noise samples, which are easier for the model to recover, while still receiving supervision from hard samples.

Different from BD3-LM (Arriola et al., 2025), which modifies the sampled noise range, our time-aware re-weighting directly increases the relative importance of low-noise samples by adjusting their loss coefficients during continual pretraining. In the subsequent SFT stage, we adopt the noise-range sampling strategy of BD3-LM. The two methods therefore operate through different mechanisms and can be used complementarily.

3.3 Supervised Fine-Tuning with Block Treatment

Following the continual pre-training phase, we proceed to supervised fine-tuning (SFT) to significantly enhance the model’s instruction-following capabilities. In this stage, we adopt the blockwise training paradigm proposed by Arriola et al. (2025). Unlike standard diffusion models limited by bidirectional attention, this method supports KV caching, boosting both inference speed and generative quality.

The overall SFT process is illustrated in Figure 1, Stage 2. The input sequence is segmented into multiple distinct blocks. Within each block, the model

operates as a standard diffusion language model, utilizing full bidirectional attention for inference. Conversely, the interaction between blocks is governed by an autoregressive constraint, ensuring that subsequent blocks can only attend to the results of preceding blocks. Formally, we employ a block-wise variant of the MDM loss, defined as:

$$\mathcal{L}_{\text{BD}}(x; \theta) := \sum_{b=1}^B \mathcal{L}(x^b, x^{<b}; \theta), \quad (4)$$

where $\mathcal{L}$ represents the standard MDM loss computed within block b conditioned on the history $x^{<b}$. Additionally, we incorporate an auxiliary balance loss (Wang et al., 2024a) to ensure balanced utilization among experts. To enable parallel training across all blocks for improved efficiency, we follow the approach of previous work (Arriola et al., 2025) by prepending clean tokens to the sequence. By employing a specialized attention mask, the i -th block is conditioned on the ground-truth content of all preceding blocks, thereby facilitating blockwise parallelization during training.

Aligning training with inference at the terminal block. A subtle inconsistency arises at the last block of each sequence. During training, if a sequence does not align with the block boundary, the terminal block is conventionally filled with [PAD] tokens so that all blocks have the same length and can be batched. At inference time, however, the model has not yet decided what to generate at these positions and they are filled with [MASK] tokens, resulting in a distribution gap between training and inference. An intuitive solution is to mirror inference by filling the terminal block with [MASK] tokens during training as well. However, despite its intuitiveness, we find that this approach introduces a conflict with the prior pre-training paradigm, subsequently degrading generation quality on short answers (Section 5.3). Therefore, we instead propose to fill the incomplete terminal block with [EOS] tokens before applying the standard random masking (Figure 1, Stage 2). This ensures that the [EOS] tokens participate in training under the same masking distribution as the rest of the sequence. The loss is computed over all original response tokens together with the appended [EOS] tokens, teaching the model when and how to terminate. We apply the same treatment into RL (Section 3.4) as well.

3.4 Block-oriented Reinforcement Learning

We adopt the trajectory-aware RL framework of TraceRL (Wang et al., 2025), which optimizes the policy along the actual denoising trajectory produced at inference time rather than over randomly masked sequences. We integrate this framework into our pipeline with three refinements that address issues specific to our setting: a train-inference alignment for terminal blocks (already introduced in Section 3.3), a better sampling strategy, and a packed-trajectory scoring scheme.

The specific process is illustrated in Figure 1, Stage 3. Following the definitions in previous works, for each input $q \in D_{\text{task}}$ from the training dataset D_{task} , we define $\{\tau_i\} \triangleq (\tau_i(1), \tau_i(2), \dots, \tau_i(T))$ as the trajectory in the decoding process, where T denotes the number of decoding steps, and $\tau_i(t)$ represents the set of tokens obtained at the t -th step. Based on these definitions, the overall optimization objective can be formulated as:

$$r_{i,t,k}(\theta) = \frac{\pi_{\theta}(o_k \mid \tau_i(1:t-1))}{\pi_{\text{old}}(o_k \mid \tau_i(1:t-1))}, \quad (5)$$

$$\mathcal{L}(\theta) = \frac{1}{G} \sum_{i=1}^G \frac{1}{N} \sum_{t=1}^T \sum_{o_k \in \tau_{i,t}} C_{\epsilon}(r_{i,t,k}(\theta), A_i), \quad (6)$$

$$\mathcal{J}(\theta_p) = \mathbb{E}_{\rho}[\mathcal{L}(\theta)], \quad (7)$$

where ρ denotes the joint distribution $q \sim D_{\text{task}}$, $\{\tau_i\}_{i=1}^G \sim \pi_{\text{old}}(\cdot \mid q)$, $C_{\epsilon}(r, A) \triangleq \min(rA, \text{clip}(r, 1 - \epsilon, 1 + \epsilon)A)$ is the clipped surrogate reward function, $\tau_i(1:t) \triangleq \bigcup_{j=1}^t \tau_i(j)$ denotes the collection of tokens up to time t , A_i is the advantage function, and β is the regularization parameter. By utilizing this optimization objective $\mathcal{J}_{\text{policy}}(\theta_p)$, we can perform training directly using RLVR (Shao et al., 2024). We also adopt clipping and dynamic sampling from DAPO (Yu et al., 2025) to stabilize training, and we train the policy directly without a separate value model.

Following Wang et al. (2025), we produce each rollout step by denoising a single token at a time, which they show yields higher-quality trajectories than parallel multi-token denoising. To increase exploration during RL, we additionally apply temperature, top- p , and top- k sampling on the per-step distribution, similar to AR decoding. This recovers the diversity and controllability of AR-style sampling while remaining within the DLLM denoising framework.

A direct implementation of trajectory-aware scoring requires one forward pass per rollout step, since each step exposes a different set of clean tokens to the model. We eliminate this redundancy by packing all rollout steps of one trajectory into a single sequence. Concretely, for a response of N tokens generated over N steps, instead of forming N separate samples: $[q, \text{MMMM}], [q, \text{M}x_2\text{MM}], \dots, [q, \text{M}x_2x_3x_4]$, we concatenate them into a single sequence: $[q, \text{MMMM}, \text{M}x_2\text{MM}, \dots, \text{M}x_2x_3x_4]$, and use a structural attention mask that (i) lets every step’s tokens attend to the shared prompt and clean tokens q , and (ii) isolates the steps from one another so that gradients computed at each step depend only on its own context. This reduces per-trajectory scoring to a single forward pass and shares the prompt and clean-prefix computation across all rollout steps.

3.5 Blockwise Inference

We follow the blockwise masked diffusion strategy of [Arriola et al. \(2025\)](#) for inference. The sequence is split into blocks: tokens within a block are decoded in parallel with bidirectional attention, while blocks are generated one after another in autoregressive order. Within each block, we decode in parallel every position whose predicted confidence is above a threshold τ . If no position is confident enough, we fall back to decoding the token with the highest confidence. Building on this, we notice that the boundary between two blocks is a useful point that AR models do not have: it is a clean break where one block’s parallel decoding has just finished and the next block has not yet started. We take advantage of this in two ways, covering KV cache management ([Li et al., 2024b](#)), and request scheduling.

Inline block prefill. In a standard blockwise pipeline, after a block is fully denoised, an additional forward pass is performed on its clean tokens to materialize their final KV states in the cache before the next block starts denoising. Instead of treating this as a standalone cache-building pass, we fold it into the first denoising step of the next block. Concretely, once block b is finalized, the next forward pass takes the clean tokens of block b together with the initial noisy tokens of block $b+1$: the clean tokens of block b are used to compute and write their final KV states into the cache, while the same forward pass also produces the prediction

for the first denoising step of block $b+1$. Thus, the prefill of block b is performed inline with the first denoising step of block $b+1$, removing the dedicated cache-building pass between blocks without changing the resulting computation.

Block-level scheduling. For AR models, sequence-level scheduling (e.g., continuous batching ([Kwon et al., 2023](#))) dynamically manages each sequence’s generation, maximizing system throughput by avoiding head-of-line blocking from slower requests. This strategy does not directly transfer to blockwise DLLMs: generation proceeds block by block, and the number of denoising steps a request needs to finish a given block depends on how quickly its intra-block confidence threshold is met. Under standard batched scheduling, faster requests must wait for the slowest one in the batch to finish the current block before any of them can advance, leaving compute idle. We adapt sequence-level scheduling to the block level: as soon as a request finishes its current block, its KV cache is collected and the request is dispatched into the next block’s compute stream, independently of other requests in the batch. This recovers the throughput benefits of sequence-level scheduling within the blockwise DLLM setting, and the effect grows with batch size, since larger batches exhibit greater variance in per-block step counts and therefore offer more idle time to recover.

Specifically, sequence-level scheduling operates at the request level: a request leaves the batch once it has produced all of its tokens. Within a request, however, the generation of individual blocks is not scheduled. Consider two requests with output lengths of 16 and 32 tokens and a block size of 4. If Request 1 finishes denoising its first block while Request 2 still needs further iterations, Request 1 cannot start its second block and idles until Request 2 catches up. The batch therefore advances at the speed of its slowest member on every block, and the cost accumulates over the blocks within a sequence. Block-level scheduling removes this constraint by letting each request advance to its next block as soon as the current one is complete. Together, these designs significantly reduce inference latency without affecting generation quality.

The full pre-training, post-training and inference pipeline described above is implemented on a customized training and inference suite, extended to support masked diffusion training, blockwise inference, and trajectory-aware RL rollouts. Full

implementation details, including the engineering optimizations are provided in Appendix D.

4 Experiment

4.1 Experiment Settings

We initiate ReDLLM from the weight of Qwen3-30B-A3B¹. Continual pre-training is conducted on a comprehensive dataset spanning four distinct domains: general text, math, code, and multilingual content, totaling 220B tokens. To ensure rigorous data quality, we implement a multi-stage filtering pipeline that involves LLM-based perplexity filtering and quality scoring via a smaller classifier model. The mixing ratios for these categories are determined using DeMix (Li et al., 2026b). To maintain alignment with the initialized parameters, all text is processed using the standard Qwen3 tokenizer. The SFT corpus consists of approximately 3.8M samples, covering instruction following, math, science, reasoning, world knowledge, code, open-domain dialogue, and in-house curated data, on which we train for two epochs. The RL dataset consists of 56K samples covering mainly math, instruction-following, and code data. More training details can be found in the Appendix A.

To comprehensively analyze the performance of ReDLLM, we conduct a systematic evaluation on widely used benchmarks. The evaluation covers seven core categories: knowledge, mathematics, coding, reasoning, multilingual capabilities, domain-specific tasks, and others, which are detailed in Appendix B. We compare ReDLLM against several strong baselines, including five DLLMs: Dream-7B (Ye et al., 2025), LLaDA2.1-Mini (16B-A1B), LLaDA2.1-Flash (100B-A6B) (Bie et al., 2026), WeDLM-8B (Liu et al., 2025), and SDAR-30B-A3B (Cheng et al., 2025a). During evaluation, we use the default configuration provided by the OpenCompass framework (OpenCompass Contributors, 2023), with the temperature set to 0 to ensure stable and reproducible results. For speed benchmarking, all models are evaluated on the same GPU. Specifically, the LLaDA 2.1 models are benchmarked using an adapted version of SGLang (Bie et al., 2026), and SDAR is evaluated using LMDeploy (LMDeploy Contributors, 2023; Zhang et al., 2025).

¹<https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507>

4.2 Downstream Performance

Table 1 reports ReDLLM and existing DLLM baselines across 30 benchmarks spanning seven categories. With the full training pipeline ($\frac{1}{t}$ (re-weighting factor), SFT, and RL), ReDLLM reaches an overall average of 69.07, outperforming all DLLM baselines at the same active-parameter budget by a wide margin. The gap to LLaDA-2.1-flash, a much larger 100B-A6B DLLM, is reduced to roughly 1.3 points on the overall average, with ReDLLM matching or surpassing it on Knowledge, Coding, and Domain-specific categories despite using only 30% parameters.

Decomposing the contribution of each training stage, the three ReDLLM columns show that each stage moves the overall average forward: SFT alone already brings ReDLLM well above all same-size DLLM baselines, the $\frac{1}{t}$ re-weighted continual pre-training adds further gains particularly on Coding and Knowledge, and the RL stage delivers the largest additional jump on Math and Reasoning categories where on-policy verifiable rewards are most informative. The variation across categories reflects the composition of our RL dataset, which is dominated by mathematical reasoning tasks; correspondingly, we observe the most consistent gains in Math and Reasoning, while other categories show only minor fluctuations from the post-SFT checkpoint, which is expected behavior under task-targeted RL.

For reference, we also report the AR base model Qwen3-30B-A3B (71.50 overall), from which our model is converted. ReDLLM reaches 69.07, retaining 96.6% of the base model’s score, while SDAR, converted from the same base, reaches 63.24. The conversion therefore transfers most of the base model’s capability at the 30B MoE scale, and the gap to it is substantially smaller than that of prior AR-to-diffusion conversions such as SDAR.

4.3 Inference Speed Comparison

Table 2 reports inference throughput at batch size 64 across six evaluation tasks, against three representative open DLLM baselines: LLaDA-2.1-mini, LLaDA-2.1-flash, and SDAR. On average, ReDLLM achieves the highest throughput at 1866.75 TPS, outperforming other baselines. The improvement over LLaDA-2.1-mini is particularly notable given that ReDLLM activates $3\times$ more parameters, indicating the superiority of our inference designs. While ReDLLM consistently excels, we

Benchmark	AR Baseline	DLLM Baseline					ReDLLM (Ours)		
	<i>Instruct Model</i>	<i>Instruct Model</i>					<i>Instruct Model</i>		
	Qwen3-30B-A3B	Dream	LLaDA-mini	LLaDA-flash	WeDLM	SDAR	+SFT	+ $\frac{1}{\epsilon}$, SFT	+ $\frac{1}{\epsilon}$, SFT, RL
Model Size	30B-A3B	7B	16B-A1B	100B-A6B	8B	30B-A3B	30B-A3B		
Overall Average	71.50	51.46	62.83	70.36	63.06	63.24	67.61	68.44	<u>69.07</u>
Knowledge									
Average	75.40	51.38	61.69	73.84	67.61	66.41	73.74	73.84	<u>73.65</u>
MMLU (5-shot)	82.95	72.84	71.12	81.48	77.97	82.32	81.61	82.21	82.20
MMLU-Pro (5-shot)	75.97	46.99	53.91	73.92	58.92	44.70	69.24	69.31	69.74
GaoKao	84.44	42.50	78.50	85.61	81.07	84.61	87.67	87.76	86.44
SimpleQA	58.24	43.20	43.21	54.34	52.46	54.01	56.44	56.08	56.23
Math									
Average	72.92	51.87	69.86	74.39	64.62	60.47	67.54	67.55	<u>69.71</u>
MATH500 (0-shot)	94.00	48.40	87.60	91.40	73.20	66.00	79.80	80.40	83.60
GSM8K (4-shot)	96.06	83.09	92.95	95.30	92.04	91.05	94.24	94.77	94.84
MATH	92.58	48.64	87.82	92.00	72.02	62.36	78.82	79.72	82.32
OlympiadBench	43.66	11.31	43.28	54.75	33.88	22.66	40.45	39.12	41.84
GSM-Plus	75.81	62.09	73.17	75.99	71.98	71.40	73.89	74.11	74.76
HARDMATH2	14.69	32.70	10.90	15.64	23.22	18.48	13.74	13.74	16.59
CMATH-dev	93.67	76.83	93.33	95.67	86.00	91.33	91.83	91.00	94.00
Coding									
Average	66.60	43.89	53.64	63.73	54.95	59.81	61.92	<u>65.09</u>	65.26
MBPP	79.80	57.60	39.40	67.40	62.40	64.00	71.00	74.00	74.20
MBPP+	66.93	53.97	58.99	63.23	61.64	60.05	60.85	64.55	63.76
HumanEval	93.54	57.32	83.17	91.95	78.05	84.63	86.34	89.63	89.88
HumanEval+	89.02	51.83	79.27	84.76	71.95	75.61	80.49	84.76	85.37
Spider	31.00	27.60	32.10	36.80	31.00	39.20	37.50	38.70	38.70
NatureCodeBench	39.29	15.00	28.93	38.21	24.64	35.36	35.36	38.93	39.64
Reasoning									
Average	90.04	75.31	82.45	90.31	80.09	85.19	86.08	85.97	<u>86.43</u>
BBH (3-shot)	89.92	62.22	78.68	90.24	68.99	79.91	81.91	82.45	83.22
PIQA	91.35	86.78	84.82	90.59	87.92	91.84	89.45	89.34	89.55
DROP	88.85	76.92	83.85	90.10	83.36	83.83	86.88	86.11	86.52
Multilingual									
Average	56.26	42.07	49.79	55.93	54.29	56.79	54.05	<u>54.27</u>	54.20
Open-MT	57.43	49.16	54.33	58.24	57.81	58.92	55.06	55.59	55.35
OpenMT-flores	26.44	16.58	21.31	26.29	25.48	27.00	24.72	24.58	24.61
CMMLU (5-shot)	84.92	60.46	73.74	83.26	79.59	84.46	82.38	82.63	82.65
Domain-specific									
Average	67.53	50.87	56.41	63.88	60.08	62.76	65.75	66.25	<u>66.12</u>
SNS-Bench	57.38	50.27	49.38	55.82	50.20	50.03	53.69	53.88	53.97
EComStage	83.81	59.15	76.35	80.22	81.80	78.74	83.08	83.18	83.28
Medbullets	60.71	41.07	45.29	53.57	49.51	55.36	58.44	60.39	59.74
MedMCQA	68.20	53.00	54.63	65.89	58.79	66.91	67.77	67.56	67.49
Others									
Average	74.76	52.09	68.33	72.75	65.32	57.49	68.55	69.56	<u>70.58</u>
NewsBench	69.33	52.22	59.78	64.89	60.89	60.67	66.67	65.11	66.89
COMPASS	69.10	43.69	60.45	68.28	61.23	47.68	59.70	63.95	64.56
IFEval	85.86	60.35	84.76	85.09	73.85	64.12	79.27	79.62	80.28

Table 1: Main results on various benchmarks. Qwen3-30B-A3B-Instruct-2507 is included as an autoregressive baseline. ReDLLM outperforms most existing DLLMs and remains competitive with the autoregressive baseline. We use bold font and underlines to highlight the best/second-best average results among DLLMs in each category and overall. The autoregressive baseline is excluded from this ranking.

Model	TPS (tok/s) at batch size 64						Avg.
	MMLU	GSM8K	MATH	HumanEval+	MBPP+	IFEval	
LLaDA-mini (16B-A1B)	804.1	2225.95	1922.12	2444.79	2251.86	1147.82	1799.44
LLaDA-flash (100B-A6B)	125.32	261.73	234.47	316.32	267.53	156.92	227.05
SDAR (30B-A3B)	293.25	1980.01	2192.49	2202.86	1492.29	1923.62	1680.75
ReDLLM (30B-A3B) (ours)	523.49	2722.69	1977.40	2404.75	1663.66	1908.50	1866.75

Table 2: Inference throughput (tokens / sec) across evaluation tasks at batch size 64. Higher is better.

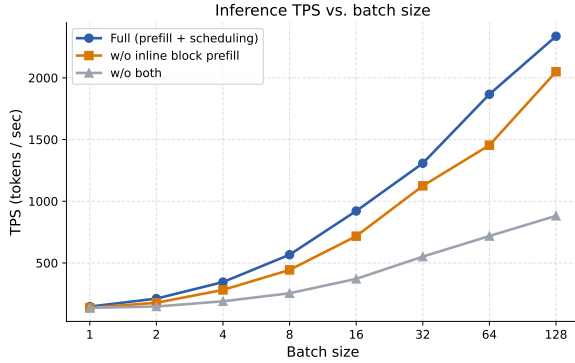


Figure 2: Ablation study for two components in ReDLLM inference. We report TPS (tokens/sec) under different batch sizes.

do observe throughput variation across different tasks. We attribute this to differences in response-length distribution and intra-block confidence dynamics, both of which affect how much idle time block-level scheduling can recover. To better understand the overall performance gains, Section 5.2 additionally analyzes the two specific designs in our inference framework.

5 Ablation Study

5.1 Comparison with TraceRL

To isolate the effect of our refinements relative to the original trajectory-aware RL framework of Wang et al. (2025), we run TraceRL on the same SFT checkpoint, with the same dataset and training budget. The two setups differ in three ways: (i) rollouts are produced by stochastic sampling with temperature, top- p , and top- k on the per-step distribution, rather than the near-greedy static sampling used by TraceRL; (ii) terminal blocks are filled with [EOS] tokens before random masking (Section 3.3), aligning training with the inference-time distribution; and (iii) per-rollout scoring is reduced into a single forward pass via packed-trajectory scoring (Section 3.4). Under this controlled comparison, ReDLLM reaches an overall average of 69.07 versus 68.55 for TraceRL, a 0.52 point improvement

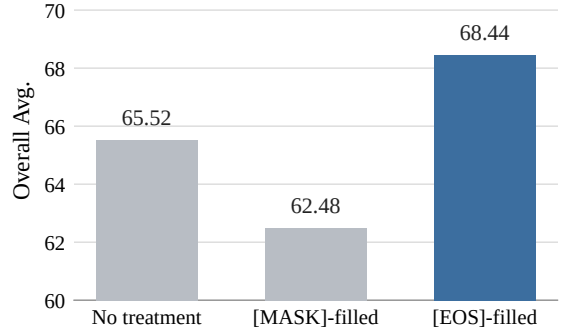


Figure 3: Comparison of three strategies for filling incomplete terminal blocks during SFT.

on top of an already strong RL framework.

5.2 Inference Strategy

Figure 2 shows throughput from batch size 1 to 128, averaged over our main evaluation tasks. The two designs together raise throughput from 881 to 2336 tokens/s at batch size 128 (a $2.65 \times$ speedup), with block-level scheduling contributing the larger share: removing inline block prefill alone drops throughput by 12%, while removing both drops it by 62%. The gap between the full method and the baseline grows with batch size, consistent with the fact that larger batches expose more idle time from cross-request variance in per-block step counts.

5.3 Terminal Block Treatment

In Section 3.3, we fill incomplete terminal blocks with [EOS] tokens before applying random masking. We now justify this design choice from two angles: benchmark performance, and failure cases.

Quantitative comparison. We train three SFT variants that are identical in every aspect except for terminal block treatment. Figure 3 reports the performance across our evaluation suite.

The “No treatment” variant stands for conventional methods, where training-inference gap remains. The [MASK] variant exactly matches what

Prompt: Question: The WWI Q-Ship “Salvia” was partially reconstructed with a false counter-stern to resemble what kind of ship?
Options:
A. merchant vessel B. cargo liner C. tramp D. fishing trawler
Answers:

[MASK]-filled: A [EOS] import [EOS]
[EOS]-filled (ours): A [EOS] [EOS] [EOS]

Prompt: Question: What did Daemon Targaryen say to Rhaenyra about living life in fear in Episode 4, Season 1 of House of the Dragon?
Options:
A. Fear will rob you of your happiness if you let it.
B. Living in fear will only bring you sorrow and regret.
C. A life lived in fear is a life half-lived, remember that.
D. You cannot live your life in fear, or you will forsake the best parts of it.
Answers:

[MASK]-filled: C [EOS] import [EOS]
[EOS]-filled (ours): C [EOS] [EOS] [EOS]

Table 3: Representative failure case of [MASK]-filled SFT on short-answer prompts. The model fills the unused tail of the terminal block with semantically incoherent tokens instead of terminating.

the model sees at inference, yet performs worse than other two methods. We examine why next.

Failure cases under [MASK] filling. Although [MASK] filling appears to be the most principled choice, in practice it produces a characteristic failure mode: on prompts that admit short answers, the model frequently generates extra tokens rather than terminating cleanly. We illustrate this with representative cases in Table 3.

6 Analysis of Expert Routing

Each MoE layer has 128 experts with top- k routing ($k = 8$), an auxiliary load-balancing loss of weight 0.001, and no explicit expert capacity constraint; the router reads only the hidden states and is not conditioned on the diffusion timestep t . Table 4 reports how this router behaves on 20 Math and 20 Code samples. No expert collapse occurs on either dataset: the normalized entropy stays close to uniform and few experts are never selected. Expert usage is more concentrated on Code, where both the max-to-mean load ratio and the load coefficient of variation are higher, which we attribute to the more structured and repetitive nature of code tokens. Because a DLLM revisits the same positions at every denoising step, we also measure the Jensen–Shannon divergence between the routing distributions of adjacent timesteps. It averages

Metric	Math	Code
Normalized entropy $\uparrow$	0.946 ± 0.016	0.897 ± 0.025
Max-to-mean load ratio $\downarrow$	4.602 ± 0.911	6.504 ± 1.737
Load CV $\downarrow$	0.760 ± 0.116	1.111 ± 0.179
Dead-expert ratio $\downarrow$	0.058 ± 0.039	0.146 ± 0.064
JSD, adjacent timesteps	0.313 ± 0.032	0.307 ± 0.015

Table 4: Expert routing statistics over 128 experts, measured on 20 samples per dataset, each with at least two diffusion timesteps. We report mean $\pm$ std across layers. The first four rows aggregate routing counts over all tokens and timesteps; under uniform routing they would be 1, 1, 0, and 0. The last row uses natural log, so JSD lies in $[0, \log 2]$ with $\log 2 \approx 0.693$.

0.313 on Math and 0.307 on Code, about 45% of the upper bound $\log 2$, so routing is neither fixed across timesteps nor rebuilt at each step; the model keeps its expert pool well utilized while moderately reallocating it as the sequence is unmasked.

7 Conclusion

We present ReDLLM, a 30B-A3B Diffusion Large Language Model trained with three refinements: time-aware weighting for continual pre-training, terminal-block treatment for post-training, and block-aware inference combining inline block pre-fill with block-level scheduling. Across extensive benchmarks, ReDLLM surpasses DLLMs at the same active-parameter scale and attains highest inference throughput among evaluated models. We hope ReDLLM serves as a practical recipe for further work on scaling diffusion language models.

Limitations

Several limitations remain in this work. First, ReDLLM is initialized from a single AR checkpoint (Qwen3-30B-A3B), which is the mainstream approach for building DLLMs, and we leave the verification of the proposed designs on from-scratch training to future work. Second, all ablations are conducted at the full 30B-A3B scale with the same data and compute budget as the main runs, which limits the granularity of more detailed design-space exploration we can afford, and we leave experiments on other model scales to future work. Finally, our evaluation focuses on text-only benchmarks; extending ReDLLM to multi-modal or long-context settings is left for future work.

Ethical Statements

ReDLLM is trained on publicly available text data and may inherit biases and factual errors present in that data. Outputs should not be used without verification in high-stakes settings such as medical, legal, or financial advice. We encourage downstream users to apply appropriate safeguards when deploying the model. We use LLM only for language polishing.

References

- Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. [Block diffusion: Interpolating between autoregressive and diffusion language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, and 31 others. 2026. [Llada2.1: Speeding up text diffusion via token editing](#). *Preprint*, arXiv:2602.08676.
- Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, and 12 others. 2025. [Llada2.0: Scaling up diffusion language models to 100b](#). *Preprint*, arXiv:2512.15745.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. [Compassjudge-1: All-in-one judge model helps model evaluation and evolution](#). *Preprint*, arXiv:2410.16256.
- Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. [Benchmarking large language models on answering and explaining challenging medical questions](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, Albuquerque, New Mexico. Association for Computational Linguistics.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. 2025a. [Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation](#). *Preprint*, arXiv:2510.06303.
- Zicong Cheng, Guo-Wei Yang, Jia Li, Zhijie Deng, Meng-Hao Guo, and Shi-Min Hu. 2025b. [Deer: Draft with diffusion, verify with autoregressive models](#). *Preprint*, arXiv:2512.15176.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. [DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Hongcheng Guo, wangyue, Shaosheng Cao, Fei zhao, Boyang Wang, Lei Li, Liang Chen, Xinze Lyu, Zhe Xu, Yao Hu, and Zhoujun Li. 2025b. [SNS-bench: Defining, building, and assessing capabilities of large language models in social networking services](#). In

- Forty-second International Conference on Machine Learning.
- Hongcheng Guo, Zheyong Xie, Shaosheng Cao, Boyang Wang, Weiting Liu, Zheyu Ye, Zhoujun Li, Zuozhu Liu, and Wei Lu. 2025c. Pet-bench: Benchmarking the abilities of large language models as e-pets in social network services. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6402–6407.
- Hongcheng Guo, Fei Zhao, Shaosheng Cao, Xinze Lyu, Ziyang Liu, Yue Wang, Boyang Wang, Zhoujun Li, Chonggang Lu, Zhe Xu, and 1 others. 2025d. Redefining machine translation on social network services with large language models. *arXiv preprint arXiv:2504.07901*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popović, and 3 others. 2024. *Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet*. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46, Miami, Florida, USA. Association for Computational Linguistics.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024a. *CMMLU: Measuring massive multitask language understanding in Chinese*. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285, Bangkok, Thailand. Association for Computational Linguistics.
- Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. 2024b. A survey on large language model acceleration based on kv cache management. *arXiv preprint arXiv:2412.19442*.
- Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. 2026a. Beyond fixed: Training-free variable-length denoising for diffusion large language models. In *International Conference on Learning Representations*, volume 2026, pages 91715–91731.
- Miao Li, Ming-Bin Chen, Bo Tang, ShengbinHou ShengbinHou, Pengyu Wang, Haiying Deng, Zhiyu Li, Feiyu Xiong, Keming Mao, Cheng Peng, and 1 others. 2024c. Newsbench: a systematic evaluation framework for assessing editorial capabilities of large language models in chinese journalism. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9993–10014.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024d. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984.
- Shengrui Li, Fei Zhao, Kaiyan Zhao, Jieying Ye, Haifeng Liu, Fangcheng Shi, Zheyong Xie, Yao Hu, and Shaosheng Cao. 2026b. Decouple searching from training: Scaling data mixing via model merging for large language model pre-training. *arXiv preprint arXiv:2602.00747*.
- Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. 2025. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The twelfth international conference on learning representations*.
- Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. 2025. *Wedlm: Reconciling diffusion language models with standard causal attention for fast inference*. Preprint, arXiv:2512.22737.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572.

- Yue Liu, Yuzhong Zhao, Zheyong Xie, Qixiang Ye, Jianbin Jiao, Yao Hu, Shaosheng Cao, and Yunfan Liu. 2026. Balancing understanding and generation in discrete diffusion models. [arXiv preprint arXiv:2602.01362](#).
- LMDeploy Contributors. 2023. Lmdeploy: A toolkit for compressing, deploying, and serving llm. <https://github.com/InternLM/lmdeploy>.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. [Scaling up masked diffusion models on text](#). In [The Thirteenth International Conference on Learning Representations](#).
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. [Large language diffusion models](#). In [The Thirty-ninth Annual Conference on Neural Information Processing Systems](#).
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. [No language left behind: Scaling human-centered machine translation](#). [Preprint, arXiv:2207.04672](#).
- OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. <https://github.com/open-compass/opencompass>.
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2026. [Your absorbing discrete diffusion secretly models the conditional distributions of clean data](#). [Preprint, arXiv:2406.03736](#).
- Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. [Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering](#). In [Proceedings of the Conference on Health, Inference, and Learning](#), volume 174 of [Proceedings of Machine Learning Research](#), pages 248–260. PMLR.
- James V Roggeveen, Erik Y Wang, David Ettel, Will Flintoft, Peter Donets, Raglan Ward, Ahmed Roman, Anton Marius Graf, Siddharth Dandavate, Ava Williamson, and 1 others. Hardmath2: A benchmark for applied mathematics built by students as part of a graduate class. In [The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track](#).
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. [arXiv preprint arXiv:2402.03300](#).
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In [Proceedings of the Twentieth European Conference on Computer Systems](#), pages 1279–1297.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. 2024. Simplified and generalized masked diffusion for discrete data. In [Proceedings of the 38th International Conference on Neural Information Processing Systems](#), pages 103131–103167.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. [arXiv preprint arXiv:1909.08053](#).
- Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, and 1 others. 2025. [Openai gpt-5 system card](#). [Preprint, arXiv:2601.03267](#).
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. [arXiv preprint arXiv:2210.09261](#).
- Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. [Efficient transformers: A survey](#). [ACM Comput. Surv.](#), 55(6).
- Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. 2024a. [Auxiliary-loss-free load balancing strategy for mixture-of-experts](#). [Preprint, arXiv:2408.15664](#).
- Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Revolutionizing reinforcement learning framework for diffusion large language models. [arXiv preprint arXiv:2509.06949](#).
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. [Advances in Neural Information Processing Systems](#), 37:95266–95290.
- Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. [arXiv preprint arXiv:2411.04368](#).
- Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. Cmath: Can your language model pass chinese elementary school math test? [arXiv preprint arXiv:2306.16636](#).

- Kaiwen Yan, Hongcheng Guo, Xuanqing Shi, Shaosheng Cao, Donglin Di, and Zhoujun Li. 2025. Codeif: Benchmarking the instruction-following capabilities of large language models for code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 1272–1286.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.
- Li Zhang, Youhe Jiang, Guoliang He, Xin Chen, Han Lv, Qian Yao, Fangcheng Fu, and Kai Chen. 2025. Efficient mixed-precision large language model inference with turbomind. *arXiv preprint arXiv:2508.15601*.
- Shudan Zhang, Hanlin Zhao, Xiao Liu, Qinkai Zheng, Zehan Qi, Xiaotao Gu, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024a. NaturalCodeBench: Examining coding performance mismatch on HumanEval and natural user prompts. *arXiv preprint arXiv:2405.04520*.
- Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2024b. *Evaluating the performance of large language models on gaokao benchmark*. Preprint, *arXiv:2305.12474*.
- Zihou Zhang, Zheyong Xie, Li Zhong, Haifeng Liu, Yao Hu, and Shaosheng Cao. 2026. One token is enough: Improving diffusion language models with a sink token. *arXiv preprint arXiv:2601.19657*.
- Fei Zhao, Chonggang Lu, Haofu Qian, Fangcheng Shi, Zijie Meng, Jianzhao Huang, Xu Tang, Zheyong Xie, Zheyu Ye, Zhe Xu, and 1 others. 2025a. Redone 2.0: Rethinking domain-specific llm post-training in social networking services. *arXiv preprint arXiv:2511.07070*.
- Fei Zhao, Chonggang Lu, Yue Wang, Zheyong Xie, Ziyang Liu, Haofu Qian, Jianzhao Huang, Fangcheng Shi, Zijie Meng, Hongcheng Guo, and 1 others. 2025b. Redone: Revealing domain-specific llm post-training in social networking services. *arXiv preprint arXiv:2507.10605*.
- Kaiyan Zhao, Zijie Meng, Zheyong Xie, Jin Duan, Yao Hu, Zuozhu Liu, and Shaosheng Cao. 2026a. Ecomstage: Stage-wise and orientation-specific benchmarking for large language models in e-commerce. *arXiv preprint arXiv:2601.02752*.
- Kaiyan Zhao, Zheyong Xie, Zhongtao Miao, Xinze Lyu, Yao Hu, and Shaosheng Cao. 2026b. Benchmarking machine translation on chinese social media texts. *arXiv preprint arXiv:2601.22931*.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Experimental Settings

Continual Pre-training. We use AdamW with a learning rate of 1×10^{-5} and a global batch size of 8M tokens, following a warm-up followed by a constant learning rate schedule. The training is conducted on RED-Megatron with a sequence length of 2048.

Supervised Fine-Tuning (SFT). For the training setup, we initialize the model using the best-performing checkpoint from the continual pre-training stage. We employ a global batch size of 256 and a cosine annealing learning rate schedule, decaying from 1×10^{-5} down to 3×10^{-7} . We set the diffusion block size to 4 to achieve an optimal trade-off between generation quality and inference speed.

Reinforcement Learning (RL). For the RL stage, we trained the model using a learning rate of 1×10^{-6} and a batch size of 64.

Inference. We set batch size to 64 in Table 2 to ensure that all evaluated models, except for LLaDA-2.1-flash, fit on a single GPU under the same hardware budget. The confidence threshold is set to 0.8.

B Evaluation Benchmarks

To rigorously assess the capabilities of ReDLLM across diverse dimensions, we adopt a comprehensive suite of publicly available benchmarks spanning seven core categories. These benchmarks are

carefully selected to cover a wide spectrum of tasks, ranging from general knowledge understanding and complex mathematical reasoning to code generation, logical inference, multilingual proficiency, and domain-specific applications. Together, they provide a holistic view of the model’s strengths and limitations under varied evaluation settings. The detailed list of benchmarks used in each category is presented below:

- **Knowledge:** MMLU (Hendrycks et al., 2020), MMLU-Pro (Wang et al., 2024b), GaoKao (Zhang et al., 2024b), SimpleQA² (Wei et al., 2024);
- **Math:** MATH500 (Lightman et al., 2021), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), Olympiad-Bench (He et al., 2024), GSM-Plus (Li et al., 2024d), HARDMath2 (Roggeveen et al., 2023), CMath (Wei et al., 2023);
- **Coding:** MBPP (Austin et al., 2021b), MBPP+ (Liu et al., 2023), HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023), Spider (Yu et al., 2018), Natural-CodeBench (Zhang et al., 2024a), Codeif (Yan et al., 2025);
- **Reasoning:** BBH (Suzgun et al., 2022), PIQA (Bisk et al., 2020), DROP (Dua et al., 2019);
- **Multilingual:** OpenMT (Kocmi et al., 2024), OpenMT-flores (NLLB Team et al., 2022), CMMLU (Li et al., 2024a), CSM-MTBench (Zhao et al., 2026b), RedTrans (Guo et al., 2025d);
- **Domain-Specific:** SNS-Bench (Guo et al., 2025b), EcomStage (Zhao et al., 2026a), Medbullets (Chen et al., 2025), MedmcQA (Pal et al., 2022), Pet-bench (Guo et al., 2025c);
- **Others:** Newsbench (Li et al., 2024c), COM-PASS (Cao et al., 2024), IFEval (Zhou et al., 2023).

All evaluations are conducted with a single run and are conducted under identical settings for all models, so the comparison remains controlled.

²<https://huggingface.co/datasets/alibaba-pai/SimpleQA-Bench>

C Additional Benchmarks

Beyond the more common and general benchmarks in Table 1, we evaluate on four additional benchmarks targeting narrower application settings in Tabel 5, including domain-specific machine translation (Redtrans and CSM-MTBench), instruction following on code (CodeIF) and LLM as virtual pet (PetBench). We report them separately because they measure specific domains rather than general ability, and mixing them into the main average would make that average harder to interpret. REDLLM is competitive on all four, reaching comparable performance with LLaDA-flash.

D ReDLLM Training Framework

To fully support the DLLM pipeline, we develop a specialized full-stack framework comprising three core components.

D.1 ReD-Megatron

Serving as the foundational infrastructure, ReD-Megatron is designed to provide robust and efficient training support for DLLMs. Built upon the industrial-grade Megatron framework (Shoeybi et al., 2019), this module incorporates substantial adaptations tailored to the specific architectural characteristics of DLLMs. Specifically, we implement native support for the MDM training objective, including the time-aware re-weighting loss described in Section 3.2. To accommodate the unique masking requirements of masked diffusion models, we optimize the internal information propagation paths and reconstruct relevant operators, enabling support for complex attention patterns including full bidirectional attention and blockwise attention mechanisms. To ensure that Tensor Parallelism strategies remain effective under these complex mask structures, we implement a cross-layer mask matrix transmission mechanism, guaranteeing computational integrity and numerical stability. Through extensive precision alignment and system validation, ReD-Megatron achieves high training throughput while maintaining convergence accuracy, establishing a solid foundation for large-scale model training.

D.2 ReD-SGLang

To fully leverage the inference potential of DLLMs, we develop ReD-SGLang, an inference engine adapted from the SGLang framework (Zheng et al.,

Benchmark	DLLM Baseline					ReDLLM (Ours)		
	Dream	LLaDA-mini	LLaDA-flash	WeDLM	SDAR	+SFT	+ $\frac{1}{\tau}$, SFT	+ $\frac{1}{\tau}$, SFT, RL
<i>Average</i>	<i>40.33</i>	<i>45.65</i>	<i>51.21</i>	<i>48.61</i>	<i>49.92</i>	<i>50.20</i>	<i>56.34</i>	<u><i>50.73</i></u>
CodeIF	18.39	34.76	42.24	33.19	33.99	36.67	36.85	38.97
CSM-MTBench	64.96	64.57	73.75	73.28	74.77	76.26	76.31	76.28
RedTrans	51.21	58.31	61.82	60.94	62.41	60.18	60.22	59.96
PetBench	26.74	24.94	27.03	27.03	28.51	27.66	27.65	27.69

Table 5: Results on four additional benchmarks targeting narrower application settings. We report them separately and exclude them from the main average. All models are evaluated under the same protocol as Table 1.

2024). As introduced in Section 3.5, ReDLLM operates on a blockwise generation mechanism. We perform comprehensive optimizations on the inference side to align with these characteristics.

First, we optimize the KV Cache workflow to enable computation overlap. In standard procedures, completing a block typically requires an additional computation step to update the KV Cache and save tokens, which introduces latency. To mitigate this, we adopt the inline block prefill. Specifically, as the current block finishes generation, its cache computation is executed simultaneously with the initial inference of the subsequent block. This pipelined overlap strategy effectively eliminates the extra time overhead between blocks. Furthermore, to address speed variance across requests within a batch, we implement block-level scheduling. Once a request finishes its current block inference, it is immediately dispatched to the next block’s computation stream without waiting for other requests in the batch.

D.3 ReD-Verl

To further enhance the reasoning capabilities of DLLMs, we develop ReD-Verl, a dedicated Reinforcement Learning (RL) Module. This module integrates the training backend of ReD-Megatron with the rollout backend powered by ReD-SGLang, combining them with the established RL training paradigms of the Verl framework (Sheng et al., 2025) to create a closed-loop system for training and generation. To support efficient policy probability estimation, we customize the inference backend to enable comprehensive recording and transmission of intermediate inference trajectories. This modification ensures that policy evaluation algorithms can obtain accurate state information and probability distributions directly from a single inference pass, thereby significantly improving the efficiency of data collection and policy updates during the reinforcement learning process.

In summary, the integration of ReD-Megatron, ReD-SGLang, and ReD-Verl provides a unified and extensible infrastructure for ReDLLM. It supports end-to-end workflows spanning continual pre-training, supervised fine-tuning, reinforcement learning, and efficient inference, enabling fast iteration and robust deployment.