# DARE: Diffusion Large Language Models Alignment and Reinforcement Executor

**Jingyi Yang*** **& Yuxian Jiang*** **& XuHao Hu***
Shanghai Artificial Intelligence Laboratory
Fudan University
yangjingyi946@gmail.com

**Shuang Cheng**
Shanghai Artificial Intelligence Laboratory
Zhejiang University

**Biqing Qi & Jing Shao†**
Shanghai AI Laboratory

**Code:** https://github.com/yjyddq/DARE

## Abstract

Diffusion large language models (dLLMs) are emerging as a compelling alternative to dominant autoregressive models, replacing strictly sequential token generation with iterative denoising and parallel generation dynamics. However, their open-source ecosystem remains fragmented across model families and, in particular, across post-training pipelines, where reinforcement learning objectives, rollout implementations and evaluation scripts are often released as paper-specific codebases. This fragmentation slows research iteration, raises the engineering burden of reproduction, and makes fair comparison across algorithms difficult. We present **DARE** (**d**LLMs **A**lignment and **R**einforcement **E**xecutor), an open framework for the post-training and evaluation of dLLMs. Built on top of verl Sheng et al. (2024) and OpenCompass Contributors (2023b), DARE unifies supervised fine-tuning, parameter-efficient fine-tuning, preference optimization, and dLLM-specific reinforcement learning under a shared execution stack for both masked and block diffusion language models. Across representative model families including LLaDA, Dream, SDAR, and LLaDA2.x, DARE provides broad algorithmic coverage, reproducible benchmark evaluation, and practical acceleration. Extensive empirical results demonstrate that DARE serves as a reusable research substrate for developing, comparing, and deploying post-training methods for current and emerging dLLMs.

## 1 Introduction

Diffusion large language models (dLLMs) Nie et al. (2025); Ye et al. (2025); Cheng et al. (2025); Khanna et al. (2025); DeepMind (2025); Song et al. (2025) have rapidly evolved from an intriguing alternative to autoregressive language modeling into a growing family of practical model architectures. Early masked diffusion large language models such as LLaDA Nie et al. (2025) and Dream Ye et al. (2025), as well as recent block diffusion or semi-autoregressive variants such as SDAR Cheng et al. (2025) and LLaDA2.x Bie et al. (2025; 2026), show that diffusion-style generation can support flexible token order, bidirectional conditioning, and parallelism. As these models mature, however, the main bottleneck is shifting away from model definition and toward post-training and evaluation infrastructure.

The current open-source dLLM ecosystem is highly fragmented. Most dLLM-based reinforcement learning (RL) methods are released as paper-specific repositories, each with its own model fork, rollout implementation, reward interface, and evaluation scripts. This

*Equal contribution
†Corresponding author

fragmentation creates at least three problems. First, it slows research iteration because integrating a new model or objective may require re-implementing the surrounding infrastructure. Second, it makes cross-paper comparison unreliable because algorithmic differences are entangled with execution and evaluation differences. Third, it raises the engineering barrier for researchers who want to study, use and extend prior work. This systems gap is becoming more important, not less. Importantly, it cannot be solved by directly reusing LLM RL frameworks. Most LLM post-training pipelines assume left-to-right generation, sequence log-probabilities over a single decoding trajectory, and rollout engines built for autoregressive serving. dLLMs instead require diffusion-aware forward and reverse processes, denoising-state likelihood surrogates, and model-family-specific rollout backends.

To address this need, we present **DARE** (**d**LLMs **A**lignment and **R**einforcement **E**xecutor), a unified post-training and evaluation framework for diffusion large language models. DARE is built on top of verl Sheng et al. (2024) for training and OpenCompass Contributors (2023b) for evaluation, while adding the missing dLLM-specific execution layers or modules. It integrates a broad set of post-training recipes, including supervised fine-tuning (SFT), parameter-efficient fine-tuning (PEFT), preference optimization, and multiple dLLM-specific RL algorithms, and exposes them through a shared pipeline. The framework supports both masked diffusion language models (MDLMs) and block diffusion language models (BDLMs), enabling one infrastructure stack to serve LLaDA Nie et al. (2025), Dream Ye et al. (2025), SDAR Cheng et al. (2025), LLaDA-MoE Zhu et al. (2025b), and LLaDA2.x Bie et al. (2025; 2026) families.

DARE also treats systems optimization as a first-class part of dLLM post-training. For MDLMs, the framework decouples the attention backend used during rollout from the one used during actor updates: rollout uses Fast-dLLM Wu et al. (2025) together with FlashAttention backends Dao et al. (2022); Dao (2023), while training uses variable-length backends of FlashAttention to reduce padding overhead. For BDLMs, DARE integrates LMDeploy Contributors (2023a); Zhang et al. (2025), SGLang Contributors (2024b), and fused loss kernels for acceleration, and integrates FlexAttention Dong et al. (2024) for training. In addition, DARE extends OpenCompass Contributors (2023b) with dLLM-aware evaluation so that benchmarking is part of the same reusable framework. DARE is not proposed at the method-level. Instead, it is a reusable research substrate that makes it possible to integrate, compare, and evaluate rapidly growing dLLM post-training methods within a unified execution environment.

Our contributions are four-fold:

- We present DARE, a unified framework that brings together dLLM post-training and evaluation under a single open-source stack, spanning masked and block diffusion language model families.
- We integrate a broad range of post-training methods, including SFT, PEFT, preference optimization, and multiple dLLM-tailored RL algorithms, enabling fairer algorithm comparison.
- We implement extensive dLLM-specific system optimizations for rollout, training, and evaluation.
- We empirically show that DARE provides broad model and benchmark coverage while turning fragmented paper-specific implementations into a more reproducible and comparable dLLM research workflow.

## 2 Related Work

### 2.1 Diffusion Language Models

**Masked Diffusion Language Models.** D3PM Austin et al. (2021a) formulates the discrete diffusion by modeling a sequence of corruption processes over categorical variables, for example with an absorbing state (<MASK>) or uniform noise Austin et al. (2021a); Wu et al. (2023); Sahoo et al. (2024); Lou et al. (2023); Zheng et al. (2024); Gong et al. (2024); Ou et al.

(2024); Nie et al. (2024). For masked diffusion language models, the categorical distribution $q(x_t^\ell | x_0^\ell)$ is parameterized by a linear interpolation between the original one-hot vector $\mathbf{x}_0^\ell$ and the absorbing vector $\mathbf{m}$ (<MASK> token):

$$q(x_t^\ell | x_0^\ell) = \mathrm{Cat}(x_t^\ell; \alpha_t \mathbf{x}_0^\ell + (1 - \alpha_t)\mathbf{m}), \tag{1}$$

where $\alpha_t$ denotes the noise schedule at timestep $t$. The mask predictor $p_\theta$ is trained to reverse the corruption process by minimizing the negative evidence lower bound objective (NELBO):

$$\mathcal{L}_\theta = \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim q(x_t|x_0), t \sim \mathcal{U}(0,1]} \left[ -\frac{1}{t} \sum_{\ell=1}^{L} \mathbb{I}[x_t^\ell = \text{<MASK>}] \log p_\theta(x_0^\ell | x_t) \right]. \tag{2}$$

Recent work Nie et al. (2025); Ye et al. (2025); Khanna et al. (2025); DeepMind (2025); Song et al. (2025) has scaled this paradigm to large language modeling. These models motivate diffusion-specific rollout and training pipelines because their generation process differs fundamentally from autoregressive left-to-right decoding.

**Block Diffusion Language Models.** In parallel, block diffusion paradigms Han et al. (2022); Arriola et al. (2025); Fathi et al. (2025); Bie et al. (2025; 2026) have produced models such as SDAR Cheng et al. (2025) and LLaDA2.x Bie et al. (2025; 2026), which combine intra-block diffusion with inter-block autoregression and support variable-length generation with kv-caching. Specifically, a sequence $x$ is partitioned into $B$ contiguous, non-overlapping blocks, $\{x^1, \ldots, x^B\}$, each containing $L' = \frac{L}{B}$ tokens. The likelihood factorizes over blocks as $\log p_\theta(x) = \sum_{b=1}^{B} \log p_\theta(x^b | x^{<b})$, and the block-wise NELBO is given by:

$$\mathcal{L}_\theta = \mathbb{E}_{x_0 \sim p_{\text{data}}, b \sim \mathcal{U}[1,B], t \sim \mathcal{U}(0,1]} \left[ -\frac{1}{t} \sum_{\ell=1}^{L'} \mathbb{I}[x_t^{b,\ell} = \text{<MASK>}] \log p_\theta(x_0^{b,\ell} | x_t^b, x^{<b}) \right] \tag{3}$$

where $x_0^{b,\ell}$ and $x_t^b$ denote the clean and corrupted sequences of block $b$, respectively. The model $p_\theta(x^b | x^{<b})$ is trained to recover the clean block $x^b$ from its noisy counterpart conditioned on preceding clean blocks $x^{<b}$. This regime introduces different systems requirements from masked diffusion, especially for rollout backends, block-wise verification, and online policy updates. The existence of both MDLM and BDLM families is one reason a unified dLLM executor is needed.

### 2.2 Post-Training Frameworks and Reinforcement Learning for dLLMs

Since 2025, dLLM post-training has developed rapidly. Early work has adapted policy-gradient-based RL algorithms to masked diffusion large language models (MDLMs). For example, VRPO Zhu et al. (2025a) is designed for human preference alignment in dLLMs. d1 Zhao et al. (2025b) and Coupled-GRPO Gong et al. (2025) directly adapt GRPO Chen et al. (2025); Shao et al. (2024); Guo et al. (2025) for one-step denoising optimization. MDPO He et al. (2025) addresses the training-inference mismatch by optimizing the progressive refinement schedule used at inference. CJ-GRPO Yang et al. (2025) emphasizes consistency between rollout and optimization trajectories, while DiFFPO Zhao et al. (2025a) studies off-policy surrogate policies and joint optimization of reasoning quality and inference-time efficiency. In addition, SPG Wang et al. (2025a) and BGPO Lin et al. (2025) introduce alternative upper-lower bounds beyond ELBO for stable optimization. However, MDLMs may face exploration limitations due to fixed-length generation. Variable-length generation strategies Li et al. (2025); Yang et al. (2026) are expected to be integrated to overcome this limitation. In contrast, the block diffusion paradigm can adaptively determine the generation length, and its corresponding RL algorithms (e.g., TraceRL Wang et al. (2025b), DiRL Zhu et al. (2025c), and EBPO Bie et al. (2026)) account for semi-autoregressive properties, explicitly emphasizing the order of rollout trajectories. These developments also illustrate why LLM RL frameworks cannot be transferred to dLLMs directly. LLMs typically employ next-token prediction and are equipped with exact sequence log-probabilities and inference engines, whereas dLLMs optimize over denoising trajectories, ELBO-style or diffusion-aware surrogates, and model-specific forward/reverse processes.

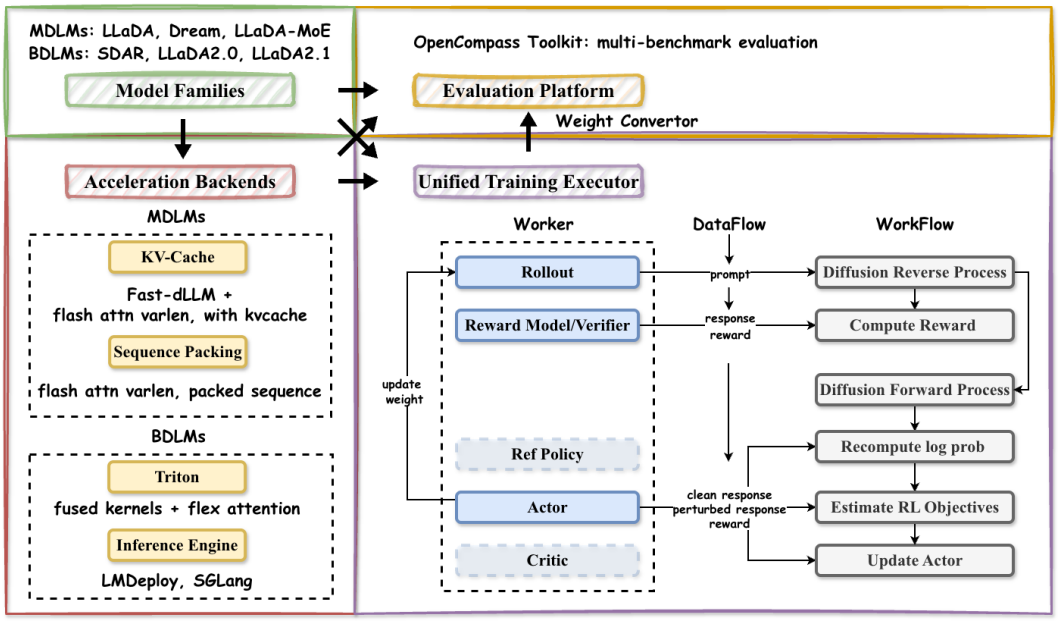



Figure 1: **High-level view of DARE.** The framework connects dLLM families, model-aware acceleration backends, a unified post-training executor, and an OpenCompass-based evaluation platform under one shared execution stack.

## 3 DARE Framework

### 3.1 Overview

DARE addresses a central systems problem in dLLM research: post-training is still fragmented across model forks, rollout implementations, reward interfaces, and benchmark evaluations. As a result, algorithms are often compared through mismatched execution pipelines rather than within a shared environment. DARE is designed as a reusable execution layer between open dLLM models, a distributed training backend, and a standardized benchmark stack. Its design follows three principles: support both masked and block diffusion models under one interface, isolate algorithm-specific logic from shared workflow, and treat evaluation as part of the framework rather than a downstream afterthought, as illustrated in Figure 1.

At the implementation level, DARE builds on verl Sheng et al. (2024) for distributed training and on OpenCompass Contributors (2023b) for evaluation, while adding dLLM-specific actor, rollout, reward, and model wrappers. The key design choice is to keep the outer execution skeleton shared and restrict customization to the truly model- or algorithm-specific parts. DARE does not introduce a new alignment objective; its systems contribution is a unified and efficient research substrate that makes previously fragmented dLLM post-training methods directly comparable under matched execution and evaluation protocols. In practice, this significantly lowers the engineering cost of integrating upcoming dLLM paradigms/algorithms and changes the practical unit of comparison from "paper plus custom codebase" to "algorithm inside a shared executor".

### 3.2 Model Families

DARE covers two dominant paradigms in current diffusion language modeling. On the masked-diffusion side, the framework supports LLaDA Nie et al. (2025), Dream Ye et al. (2025), and LLaDA-MoE Zhu et al. (2025b), whose training and rollout pipelines revolve around iterative denoising over fully visible sequences. On the block-diffusion side, DARE supports SDAR Cheng et al. (2025), LLaDA2.0 Bie et al. (2025), and LLaDA2.1 Bie et al. (2026),

Table 1: **Post-training methods integrated in DARE across supported dLLM families.** The last column indicates the original implementation or codebase from which the method was adapted, highlighting the fragmented state of prior dLLM post-training infrastructure.

| | LLaDA | Dream | SDAR | LLaDA-MoE | LLaDA2.0 | LLaDA2.1 | *Original Codebase* |
|---|---|---|---|---|---|---|---|
| **SFT/PEFT** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | model-specific |
| **DPO/VRPO** Zhu et al. (2025a) | ✓ | ✓ | | ✓ | | | closed-source |
| **d1** Zhao et al. (2025b) | ✓ | ✓ | | ✓ | | | D1 |
| **Coupled-GRPO** Gong et al. (2025) | ✓ | ✓ | | ✓ | | | Open-R1 |
| **MDPO** He et al. (2025) | ✓ | ✓ | | ✓ | | | Open-R1 |
| **CJ-GRPO** Yang et al. (2025) | ✓ | ✓ | | ✓ | | | D1 |
| **SPG** Wang et al. (2025a) | ✓ | ✓ | | ✓ | | | D1 |
| **BGPO** Lin et al. (2025) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | verl |
| **EBPO** Bie et al. (2026) | | | | | ✓ | ✓ | closed-source |

whose semi-autoregressive structure introduces different rollout order, cache behavior, and attention constraints. These model families do not merely differ in architecture; they also require different rollout backends, different training optimizations, and in some cases different policy-update paths.

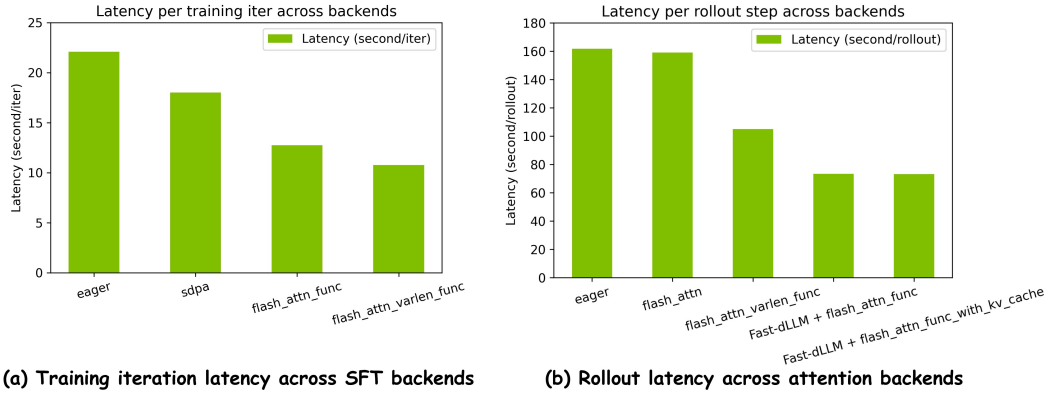
### 3.3 Unified Training Executor

DARE exposes one post-training executor covering supervised fine-tuning, parameter-efficient fine-tuning, preference optimization, and multiple dLLM-specific reinforcement learning algorithms.

**Abstractions: workers, dataflow, and workflow.** The unified executor in Figure 1 can be understood through three high-level abstractions: worker, dataflow, and workflow. ***Workers*** capture the major functional roles in post-training, including rollout, actor, reward model or verifier, and optional reference-policy and critic. ***Dataflow*** describes how prompts, responses, reward signals, perturbed trajectories, and log-probability signals move between these workers. ***Workflow*** describes the outer optimization loop itself, including rollout (reverse process), reward computation, the forward process when needed, log-probability recomputation, RL objective estimation, and actor update. This abstraction also covers SFT, PEFT, and VRPO: their recipes follow the same high-level view, but use a simplified version of the executor with fewer active components and a shorter optimization path than RL. Under this abstraction, different models, algorithms, and tasks look much more similar than their paper-specific implementations suggest. The same executor can host multiple dLLM families because what changes is usually a small number of model-aware or algorithm-aware hooks, not the outer training structure.

**Shared workflow.** DARE keeps the high-level PPO-style dataflow shared while exposing diffusion-specific customization only where it is necessary. At a coarse level, the executor reuses the same skeleton for **(i)** rollout generation, **(ii)** reward computation, **(iii)** old-policy and optional reference-policy log-probability recomputation, **(iv)** advantage or return estimation, and **(v)** actor or critic update.

**Integrated RL Algorithms.** Table 1 summarizes the RL algorithms currently available in the framework. These include VRPO Zhu et al. (2025a), d1 Zhao et al. (2025b), Coupled-GRPO Gong et al. (2025), MDPO He et al. (2025), CJ-GRPO Yang et al. (2025), SPG Wang et al. (2025a), BGPO Lin et al. (2025), and EBPO Bie et al. (2026). The last column shows that the original implementations of these RL algorithms are scattered across different codebases, whereas DARE integrates them into a unified executor. Most integrated dLLM-RL methods rely on ELBO-based, ELBO-inspired, or trace/trajectory-aware likelihood surrogates. For instance, VRPO, BGPO, SPG, and EBPO represent typical ELBO-based approaches, differing primarily in their approximation of the ELBO bound and whether they employ sequence-level or block-level formulations. In contrast, d1 and Coupled-GRPO draw upon ELBO-inspired objectives, which can be viewed as special cases of ELBO with Monte Carlo sample sizes of 1 and 2, respectively, or equivalently as one-step denoising optimization procedures. Meanwhile, MDPO and CJ-GRPO emphasize step- or trajectory-level optimization.




**(a) Training iteration latency across SFT backends** **(b) Rollout latency across attention backends**

Figure 2: **Attention backend comparison for the training and rollout stages of the masked-dLLM pipeline.** Panel (a) shows that optimized attention backends substantially reduce SFT iteration latency relative to eager execution and SDPA. Panel (b) shows that the rollout-stage backend choice has an even larger effect, with Fast-dLLM-based execution yielding the lowest rollout latency. Together, these results motivate DARE's decoupled optimization of the training and rollout paths.

**Algorithm plug-in points.** What changes across algorithms is not the entire pipeline, but a small number of well-defined hooks. Depending on the method, these hooks specify the forward corruption process, the trajectory construction rule, the likelihood estimator or bound used in optimization, and the final policy loss. Yet they still share rollout orchestration, reward dispatch, checkpointing, and evaluation. This separation is what makes side-by-side comparison meaningful: algorithm behavior can be analyzed under matched rollout and verification protocols rather than under entangled implementation differences.

### 3.4 Acceleration Backends

DARE treats acceleration as a model-aware systems problem rather than as one universal backend choice. The main reason is that training and rollout exhibit different bottlenecks, and MDLMs (bidirectional attention) and BDLMs (semi-autoregressive) introduce different execution constraints on top of that.

**Training-side optimization.** For supervised fine-tuning and log-probability recomputation, DARE focuses on reducing padding overhead and supporting longer contexts. For MDLMs such as LLaDA and Dream, the framework uses `flash_attn_varlen_func` to train efficiently on variable-length batches and combines it with sequence parallelism when scaling to longer contexts. By skipping unnecessary computation on padding tokens (e.g., `<PAD>` or `<EOS>`), this optimization significantly improves training throughput. As shown in Figure 2(a), these attention-backend optimizations translate into clear practical gains for MDLM SFT: switching from eager execution to `flash_attn_varlen_func` reduces per-iteration latency from roughly 22.1 seconds to 10.8 seconds, corresponding to an approximately 2.0$\times$ speedup. For BDLMs such as SDAR, DARE integrates FlexAttention Dong et al. (2024) to express semi-autoregressive block constraints while benefiting from compiler-driven kernel optimization. The framework also supports LoRA Hu et al. (2022) for parameter-efficient adaptation.

**Rollout-side optimization for MDLMs.** For masked diffusion models, DARE adopts a key design principle: the best rollout attention backend is not necessarily the best actor update attention backend. Therefore, we decouple the attention backend used during rollout from that used during actor updates. Rollout benefits from KV-cache reuse and repeated denoising on partially fixed states, so DARE combines Fast-dLLM Wu et al. (2025) with `flash_attn_func` or `flash_attn_with_kvcache` for fast sampling. The actor update, in contrast, operates on packed sequences with variable valid lengths, where `flash_attn_varlen_func` is more effi-

Table 2: **Baseline performance of LLaDA, Dream, SDAR, and LLaDA2.x on various benchmarks.** We use the optimal hyperparameters for each model on each benchmark, including generation length, block length, and the number of denoising steps.

| Benchmark/Model | LLaDA-8B-Instruct | Dream-7B-Instruct | SDAR-8B-Chat | SDAR-30B-A3B | LLaDA2.0-mini | LLaDA2.1-mini |
|---|---|---|---|---|---|---|
| **General QA / Reasoning** | | | | | | |
| **MMLU** | 65.24 | 66.83 | 77.23 | 79.16 | 72.54 | 69.91 |
| **MMLU-Pro** | 36.82 | 31.89 | 56.49 | 25.59 | 57.10 | 57.52 |
| **Hellaswag** | 75.30 | 63.23 | 87.59 | 92.81 | 82.35 | 78.00 |
| **ARC-C** | 87.80 | 81.36 | 86.78 | 78.98 | 85.76 | 83.39 |
| **GPQA** | 31.82 | 26.77 | 41.40 | 36.36 | 34.34 | 34.34 |
| **Mathematics** | | | | | | |
| **GSM8k** | 79.68 | 83.24 | 91.36 | 92.49 | 88.48 | 86.13 |
| **MATH** | 41.08 | 48.02 | 78.40 | 68.56 | 81.50 | 84.56 |
| **AIME24** | 2.08 | 0.83 | 13.33 | 13.33 | 16.67 | 26.67 |
| **AIME25** | 0.42 | 0.00 | 16.67 | 6.67 | 23.33 | 26.67 |
| **Olympiad** | 9.70 | 12.22 | 24.93 | 32.90 | 38.82 | 40.31 |
| **Code** | | | | | | |
| **HumanEval** | 46.34 | 78.05 | 79.88 | 84.15 | 81.10 | 81.10 |
| **MBPP** | 38.80 | 56.40 | 71.60 | 52.00 | 64.80 | 62.60 |

Table 3: **Mathematics-task results under DARE.** We report separate algorithm comparisons for LLaDA-8B-Instruct and Dream-7B-Instruct on GSM8K and MATH. Best results are in **bold**, and second-best results are underlined.

**(a) LLaDA-8B-Instruct**

| Benchmark/Algorithm | Baseline | d1 | Coupled-GRPO | VRPO | CJ-GRPO | SPG | BGPO |
|---|---|---|---|---|---|---|---|
| **GSM8K** | 76.5 | 83.7 | 85.3 | 81.9 | **85.6** | 83.5 | 82.3 |
| **MATH** | 34.6 | 40.6 | **41.0** | 35.8 | 39.2 | 40.6 | 40.0 |

**(b) Dream-7B-Instruct**

| Benchmark/Algorithm | Baseline | d1 | Coupled-GRPO | CJ-GRPO | SPG | BGPO |
|---|---|---|---|---|---|---|
| **GSM8K** | 77.2 | 82.5 | 80.3 | **85.7** | 59.4 | 83.9 |
| **MATH** | 39.6 | 49.7 | 40.4 | **50.7** | 25.2 | 48.9 |

cient because it avoids padding-heavy computation. Figure 2(b) further shows that rollout-side optimization is even more consequential: Fast-dLLM combined with `flash_attn_func` or `flash_attn_with_kvcache` reduces rollout latency from about 161.6 seconds under eager execution to about 73.4–73.5 seconds, yielding an approximately 2.2× speedup for the MDLM rollout path. By explicitly decoupling rollout and training attention backends, DARE achieves an end-to-end RL pipeline speedup of approximately 4× for the MDLM path reported in the repository.

**Rollout-side optimization for BDLMs.** For BDLMs, DARE adopts a different acceleration path that matches their semi-autoregressive structure. Rollout is accelerated with LMDeploy or SGLang Contributors (2023a); Zhang et al. (2025); Contributors (2024b), while training uses block-aware attention and model-specific fused operators such as SDAR's logits-free `fused_linear_cross_entropy`. The framework also supports online rollout-policy updates to address synchronization issues between the rollout and actor policies. Together, these design choices provide more than 14× acceleration for the supported BDLM RL pipeline.

### 3.5 Evaluation Platform

We integrate OpenCompass Contributors (2023b) into DARE to provide a unified and reproducible evaluation platform for dLLMs, and extend it with model-aware execution backends. For MDLMs, it supports Fast-dLLM-based acceleration Wu et al. (2025); for BDLMs, it supports LMDeploy and SGLang Contributors (2023a; 2024b). This avoids the common situation in which post-training code and benchmark code evolve independently and are difficult to reconcile. DARE covers benchmarks spanning general knowledge, commonsense reasoning, mathematics, olympiad-style reasoning, planning, and code generation, including MMLU Hendrycks et al. (2021), MMLU-Pro Wang et al. (2024), HellaSwag Zellers et al. (2019), ARC-C Clark et al. (2018), GSM8K Cobbe et al. (2021), MATH Lightman et al. (2023),

Table 4: **Code-task results under DARE.** We report separate algorithm comparisons for LLaDA-8B-Instruct and Dream-7B-Instruct on HumanEval and MBPP. Best results are in **bold**, and second-best results are underlined.

**(a) LLaDA-8B-Instruct**

| Benchmark/Algorithm | Baseline | d1 | Coupled-GRPO | VRPO | CJ-GRPO | SPG | BGPO |
|---|---|---|---|---|---|---|---|
| **HumanEval** | 46.9 | 47.6 | 45.1 | **52.4** | 45.1 | 48.8 | 45.1 |
| **MBPP** | 37.9 | 39.1 | 38.1 | **42.8** | 40.9 | 41.9 | 40.3 |

**(b) Dream-7B-Instruct**

| Benchmark/Algorithm | Baseline | d1 | Coupled-GRPO | CJ-GRPO | SPG | BGPO |
|---|---|---|---|---|---|---|
| **HumanEval** | 57.9 | 60.7 | **61.6** | 58.5 | 17.7 | 56.7 |
| **MBPP** | 56.2 | 56.5 | **60.3** | 57.5 | 54.4 | 58.7 |

GPQA Rein et al. (2024), AIME2024/2025 Contributors (2024a; 2025), OlympiadBench He et al. (2024), HumanEval Chen et al. (2021), and MBPP Austin et al. (2021b). Table 2 reports these benchmark results for LLaDA, Dream, SDAR, LLaDA2.0, and LLaDA2.1 reproduced inside DARE.

## 4 Empirical Results

### 4.1 Reproducibility

**Implementation Details.** We conduct all RL experiments on three tasks, i.e., math, code, and planning Ye et al. (2025), with a unified set of hyperparameters. Specifically, the rollout group size is set to 8, the block length is 32, and KL regularization is disabled by default. For methods that rely on Monte Carlo likelihood estimation, we set the number of Monte Carlo samples to 16. For mathematical reasoning, we train on the GSM8K training split Cobbe et al. (2021) together with the MATH training split Hendrycks et al.. In this task, we train for one epoch using a maximum response length of 512 and 256 diffusion steps. For code generation, we use 16K medium-difficulty problems filtered from DeepCoder Luo et al. (2025) as the training set, and adopt MBPP and HumanEval as the test sets. For planning tasks, we train for one epoch on Countdown Pan et al. (2025) and Sudoku Arel (2025) using a maximum response length of 256 and 128 diffusion steps, and adopt the same test splits as d1 Zhao et al. (2025b).

**Evaluation reproducibility.** We perform sanity checks against official model repositories, source papers, and released model-card figures to assess whether DARE preserves existing evaluation behavior. As shown in Table 5, LLaDA-8B-Instruct and SDAR-8B-Chat are generally close to their reference values (within 1.62 and 1.37 points, respectively), whereas Dream-7B-Instruct exhibits larger discrepancies on several benchmarks. The latter comparisons rely on an official blog figure rather than a fully specified evaluation artifact, so they should be interpreted as approximate checks rather than exact reproduction targets.

**Post-training reproducibility.** Table 6(b) compares DARE's LLaDA-8B-Instruct results with numbers reported by the corresponding RL method papers. Most reproduced results remain in a comparable range, while exact agreement is not expected because benchmark variants, decoding configurations, and evaluation protocols differ across papers. The table therefore serves as a sanity check of trends and implementation behavior rather than a claim of strict numerical identity.

**Runtime and throughput.** Under the matched settings described in Table 6(a), DARE reduces rollout time from 107.59 seconds to 70.97 seconds relative to the BGPO codebase and from 1839.6 seconds to 70.97 seconds relative to the d1 codebase. The same pattern appears in training-step time and throughput, providing direct evidence that DARE contributes practical systems efficiency in addition to framework unification.

Table 5: Evaluation sanity checks. DARE is compared with the official repository (LLaDA), original paper (SDAR), or official blog figure (Dream). Δ is DARE minus the reference.

| Model | Benchmark | DARE | Ref. | Δ |
|---|---|---|---|---|
| LLaDA-8B-Instruct | MMLU | 65.24 | 65.4 | -0.16 |
| | MMLU-Pro | 36.82 | 36.6 | +0.22 |
| | Hellaswag | 75.30 | 75.3 | +0.00 |
| | ARC-C | 87.80 | 89.2 | -1.40 |
| | GPQA | 31.82 | 32.3 | -0.48 |
| | GSM8K | 79.68 | 78.9 | +0.78 |
| | MATH | 41.08 | 42.7 | -1.62 |
| | HumanEval | 46.34 | 47.0 | -0.66 |
| | MBPP | 38.80 | 39.6 | -0.80 |
| SDAR-8B-Chat | MMLU | 77.23 | 78.6 | -1.37 |
| | MMLU-Pro | 56.49 | 56.9 | -0.41 |
| | GPQA | 41.40 | 40.2 | +1.20 |
| | GSM8K | 91.36 | 91.3 | +0.06 |
| | HumanEval | 79.88 | 78.7 | +1.18 |
| | MBPP | 71.60 | 72.0 | -0.40 |
| Dream-7B-Instruct | MMLU | 66.83 | 69.5 | -2.67 |
| | Hellaswag | 63.23 | 73.3 | -10.07 |
| | ARC-C | 81.36 | 59.8 | +21.56 |
| | GPQA | 26.77 | 36.6 | -9.83 |
| | GSM8K | 83.24 | 77.2 | +6.04 |
| | MATH | 48.02 | 39.6 | +8.42 |
| | HumanEval | 78.05 | 57.9 | +20.15 |
| | MBPP | 56.40 | 56.2 | +0.20 |

Table 6: Systems evidence for DARE. (a) Runtime and throughput under matched settings. (b) Post-training reproducibility on LLaDA-8B-Instruct. For each algorithm, the upper row is DARE and the lower row is the original method paper.

**(a) Runtime and throughput.**

| Metric | DARE | BGPO | d1 |
|---|---|---|---|
| Rollout (s) | 70.97 | 107.59 | 1839.6 |
| Rollout (tok/s) | 923.5 | 609.1 | 35.6 |
| Train step (s) | 188.73 | 297.94 | 2051.8 |
| Train (tok/s) | 347.2 | 220.0 | 31.9 |

**(b) Post-training reproducibility.**

| Algorithm | Source | GSM8K | MATH | HumanEval | MBPP |
|---|---|---|---|---|---|
| d1 Zhao et al. (2025b) | DARE | 83.7 | 40.6 | 47.6 | 39.1 |
| | Original | 81.9 | 39.2 | 34.8 | 41.6 |
| VRPO Zhu et al. (2025a) | DARE | 81.9 | 35.8 | 52.4 | 42.8 |
| | Original | 83.3 | 42.6 | 52.4 | 42.8 |
| CJ-GRPO Yang et al. (2025) | DARE | 85.6 | 39.2 | 45.1 | 40.9 |
| | Original | 84.29 | 38.00 | – | – |
| SPG Wang et al. (2025a) | DARE | 83.5 | 40.6 | 48.8 | 41.9 |
| | Original | 86.1 | 41.8 | 41.5 | 50.6 |
| BGPO Lin et al. (2025) | DARE | 82.3 | 40.0 | 45.1 | 40.3 |
| | Original | 84.3 | 45.7 | 47.6 | 41.7 |

### 4.2 Main Results

We summarize the main algorithm-comparison results for LLaDA and Dream under the unified DARE executor.

**LLaDA as the backbone.** For LLaDA-8B-Instruct, different algorithms dominate in different task regimes. As shown in Table 3 (a), CJ-GRPO achieves the best GSM8K result, while Coupled-GRPO is best on MATH, with d1 and SPG remaining competitive. For the code tasks in Table 4 (a), VRPO is strongest on both HumanEval and MBPP, while SPG ranks second. For the planning tasks in Table 7, the differences between methods become even sharper, and the ranking changes again: Coupled-GRPO is the strongest on Countdown, whereas BGPO is best on Sudoku.

**Dream as the backbone.** For Dream-7B-Instruct, the preferred algorithm is again task-dependent. As shown in Table 3 (b), CJ-GRPO achieves the best results on GSM8K and MATH, with d1 and BGPO also showing strong gains over the baseline. For the code tasks in Table 4 (b), Coupled-GRPO achieves the best results on both HumanEval and MBPP, while d1 and BGPO remain competitive. At the same time, SPG is much weaker for this backbone on both math and code, suggesting that algorithm robustness can vary substantially across model families.

**Taken together, the LLaDA and Dream results indicate that there is no single algorithm with uniformly dominant performance across all tasks in our experiments.** Some methods are particularly effective for math, others for code, and others for planning; moreover, the same method can change rank when the backbone changes. This is exactly the setting in which a unified executor is most valuable: it makes the absence of a universal winner visible under matched infrastructure.

### 4.3 Empirical Findings

Beyond the main results, Figure 3 shows that d1, Coupled-GRPO, and CJ-GRPO generally produce more stable reward curves, whereas ELBO-based methods are more sensitive to

Table 7: **Planning-task results under DARE.** We report the algorithm comparison for LLaDA-8B-Instruct on Countdown and Sudoku. Best results are in **bold**, and second-best results are underlined.

| Benchmark/Algorithm | Baseline | d1 | Coupled-GRPO | VRPO | CJ-GRPO | SPG | BGPO |
|---|---|---|---|---|---|---|---|
| **Countdown** | 16.8 | 10.7 | **77.9** | 21.5 | 65.2 | 10.1 | 10.0 |
| **Sudoku** | 26.2 | 31.8 | 21.3 | 29.0 | 25.0 | 27.9 | **42.6** |

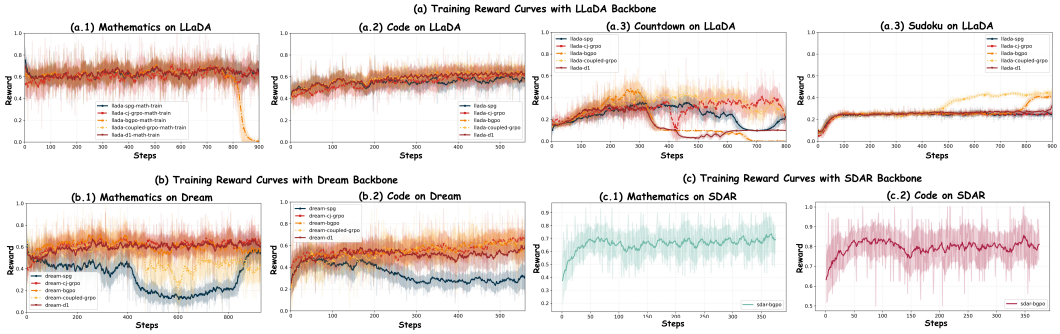



Figure 3: **Training curves across different tasks, algorithms, and backbones.**

Monte Carlo estimator noise. Representative failures include late-stage BGPO collapse on LLaDA mathematics, instability of d1/SPG/BGPO on Countdown, and unstable SPG training with Dream. These curves make the systems trade-offs visible: more precise, stable, and efficient ELBO-objective-based RL algorithms remain valuable directions. Combined with Section 4.2, the evidence favors Coupled-GRPO and CJ-GRPO as strong effectiveness–stability compromises, with d1 as a robust baseline.

## 5 Conclusion and Future Work

We presented DARE, a unified post-training and evaluation framework for diffusion large language models. Rather than contributing a single new optimization objective, DARE provides a shared execution environment that unifies model families, post-training methods, rollout implementations, reward interfaces, and evaluation pipelines under one reusable stack. Our experiments show that this framework perspective is useful in practice. DARE supports broad model and benchmark coverage and enables direct comparison of heterogeneous post-training algorithms inside one executor. There are several natural directions for future work. On the model side, diffusion vision-language and omni-language models will require integration. On the algorithm side, the executor should continue absorbing new estimators, control policies, and stability techniques. On the systems side, more comprehensive efficiency ablations and deployment-oriented evaluation backends would further strengthen the framework. We aim to keep DARE aligned with state-of-the-art diffusion large language models and post-training methods, and we welcome contributions and feedback from the open-source community.

## References

Arel. Arel's sudoku generator. https://www.ocf.berkeley.edu/ arel/sudoku/main.html, 2025.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021a.

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

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

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

Guanxu Chen, Yafu Li, Yuxian Jiang, Chen Qian, Qihan Ren, Jingyi Yang, Yu Cheng, Dongrui Liu, and Jing Shao. Conditional advantage estimation for reinforcement learning in large reasoning models. *arXiv preprint arXiv:2509.23962*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

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

AIME 2024 Contributors. Aime2024. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024a.

AIME 2025 Contributors. Aime2025. https://huggingface.co/datasets/math-ai/aime25, 2025.

LMDeploy Contributors. Lmdeploy: A toolkit for compressing, deploying, and serving llm. https://github.com/InternLM/lmdeploy, 2023a.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023b.

SGLang Contributors. Sglang. https://github.com/sgl-project/sglang, 2024b.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Google DeepMind. Gemini diffusion: our state-of-the-art experimental text diffusion model. *URL https://deepmind.google/models/gemini-diffusion/*, 2025.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

Nima Fathi, Torsten Scholak, and Pierre-André Noël. Unifying autoregressive and diffusion-based sequence generation. *arXiv preprint arXiv:2504.06416*, 2025.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*, 2024.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. Mdpo: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *Sort*, 2(4):0–6.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *arXiv preprint arXiv:2508.00819*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient rl of diffusion large language models. *arXiv preprint arXiv:2510.11683*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level. *Notion Blog*, 1, 2025.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. Spg: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025a.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025b.

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

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

Jingyi Yang, Guanxu Chen, Xuhao Hu, and Jing Shao. Taming masked diffusion language models via consistency trajectory reinforcement learning with fewer decoding step. *arXiv preprint arXiv:2509.23924*, 2025.

Jingyi Yang, Yuxian Jiang, and Jing Shao. $\rho$-EOS: Training-free bidirectional variable-length control for masked diffusion llms. *arXiv preprint arXiv:2601.22527*, 2026.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Li Zhang, Youhe Jiang, Guoliang He, Xin Chen, Han Lv, Qian Yao, Fangcheng Fu, and Kai Chen. Efficient mixed-precision large language model inference with turbomind. *arXiv preprint arXiv:2508.15601*, 2025.

Hanyang Zhao, Dawen Liang, Wenpin Tang, David Yao, and Nathan Kallus. Diffpo: Training diffusion llms to reason fast and furious via reinforcement learning. *arXiv preprint arXiv:2510.02212*, 2025a.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025b.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

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

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

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyang He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. Dirl: An efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*, 2025c.