LLaDA2.0: Scaling Up Diffusion Language Models to 100B

Tiwei Bie[1], Maosong Cao[1], Kun Chen[1], Lun Du[1], Mingliang Gong[1], Zhuochen Gong[1], Yanmei Gu[1], Jiaqi Hu[1,3], Zenan Huang[1], Zhenzhong Lan[1,4,†], Chengxi Li[1], Chongxuan Li[2], Jianguo Li[1,†], Zehuan Li[1], Huabin Liu[1], Ling Liu[1], Guoshan Lu[1], Xiaocheng Lu[1,5], Yuxin Ma[1], Jianfeng Tan[1], Lanning Wei[1], Ji-Rong Wen[2], Yipeng Xing[1], Xiaolu Zhang[1], Junbo Zhao[1,3,†], Da Zheng[1,†], Jun Zhou[1], Junlin Zhou[1], Zhanchao Zhou[1,4], Liwang Zhu[1], Yihong Zhuang[1]

[1]Ant Group, [2]Renmin University of China, [3]Zhejiang University,
[4]Westlake University, [5]HongKong University of Science and Technology

Abstract

This paper presents LLaDA2.0 — a tuple of discrete diffusion large language models (dLLM) scaling up to 100B total parameters through systematic conversion from auto-regressive (AR) models — establishing a new paradigm for frontier-scale deployment. Instead of costly training from scratch, LLaDA2.0 upholds knowledge inheritance, progressive adaption and efficiency-aware design principle, and seamless converts a pre-trained AR model into dLLM with a novel 3-phase block-level WSD based training scheme: progressive increasing block-size in block diffusion (warm-up), large-scale full-sequence diffusion (stable) and reverting back to compact-size block diffusion (decay). Along with post-training alignment with SFT and DPO, we obtain LLaDA2.0-mini (16B) and LLaDA2.0-flash (100B), two instruction-tuned Mixture-of-Experts (MoE) variants optimized for practical deployment. By preserving the advantages of parallel decoding, these models deliver superior performance and efficiency at the frontier scale. Both models were open-sourced.

Huggingface: https://hf.co/collections/inclusionAI/llada-20

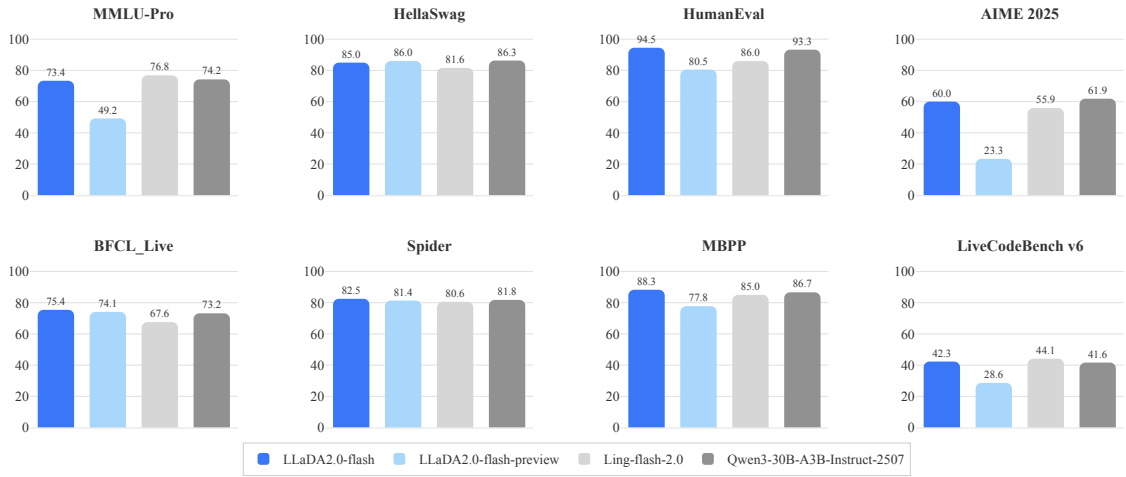

Figure 1: `LLaDA2.0-flash` main results.

Authors are listed in alphabetical order based on last name. † indicates tech-leaders.

1 Introduction

Large Language Models have achieved remarkable success through the AR paradigm, modeling sequences via next-token prediction with strict left-to-right causal dependencies (Hurst et al., 2024; Grattafiori et al., 2024; Yang et al., 2025). This approach naturally aligns with the sequential structure of language and enables efficient training through next-token likelihood maximization. However, the very success of this paradigm creates fundamental limitations: the sequential generation process imposes severe inference bottlenecks, precluding parallelization, and increasing latency at scale, while the rigid causal structure can be suboptimal for tasks requiring bidirectional reasoning and holistic understanding.

Discrete Masked Diffusion Language Models (MDLM) have emerged as a compelling alternative to the prevailing AR paradigm. By reconstructing sequences from random masked inputs, these models inherently support parallel generation and leverage a full bidirectional context, offering a different architectural approach (Gong et al., 2025; Yu et al., 2025). Although these conceptual advantages are clear, the field is still in an early developmental stage. Current research is actively focused on key challenges, including the refinement of specialized training regimes, the design of efficient sampling strategies, the efficient inference of open-source models, and reinforcement learning for MDLM. As a result of this ongoing exploration, most existing diffusion models, including recent advancements like Block Diffusion Language Models (BDLMs) (Arriola et al., 2025), operate at a smaller scale (*e.g.*, $\leq$8B parameters). Bridging this scale difference to the hundreds of billions of parameters seen in the leading mainstream AR models is a primary frontier for enabling diffusion models to fully capture complex linguistic patterns for practical deployment.

In this work, we introduce LLaDA2.0 series with 100B/16B total parameters diffusion language models that resolves these fundamental challenges through a novel two-stage continual pre-training (CPT) paradigm. Rather than attempting to train diffusion models from scratch, we leverage existing AR checkpoints as the foundation for a systematic conversion process that preserves linguistic knowledge while introducing diffusion capabilities.

The first stage, CPT aims to transform the foundational AR model into a capable diffusion language model. However, direct conversion is challenging due to the inherent data distribution gap between left-to-right generation and bidirectional denoising. Although the BDLM formulation partially reduces this gap through blockwise masked reconstruction, it suffers from low data utilization, limiting the effective exploitation of large-scale corpora. To this end, we introduce the Warmup–Stable–Decay (WSD) strategy, smoothly bridging the AR-to-dLLM gap while substantially improving CPT efficiency. WSD gradually expands the model's receptive field to introduce diffusion-style context (*Warmup*), strengthens global denoising under full-sequence training (*Stable*), and then refines the model into an efficient blockwise structure (*Decay*). This progressive adjustment enables a stable and data-efficient transition to diffusion-based learning. Additionally, under full attention in packed training sequences, diffusion models risk forming spurious dependencies across document boundaries, leading to semantic confusion and instability in bidirectional training. To prevent such cross-document interference, we introduce a document-level attention mask that restricts self-attention within individual documents, ensuring coherent context modeling.

The second stage, Post-training for Practical Deployment, transitions the model from a raw predictive engine into a capable and efficient assistant. The random masking nature of the diffusion fine-tuning objective means any single sample provides only a partial learning signal. We address this by employing a complementary masking strategy, which ensures near-100% data utilization and accelerates convergence by guaranteeing every token contributes to the model's learning. With an efficient foundation for instruction tuning, we then align the model with human preferences by adapting modern techniques like Direct Preference Optimization (DPO)—originally designed for AR models—by reformulating the objective over the model's reconstruction loss. Beyond alignment, practical deployment hinges on inference speed. To realize the full promise of parallel decoding, which is often limited by a model's lack of predictive confidence, we incorporate an auxiliary confidence prediction loss. This trains the model to be "sharper" and more certain, unlocking aggressive and efficient parallel generation without degrading quality.

We release instruction-tuned variants for practical deployment: `LLaDA2.0-mini` (16B parameters) for resource-constrained applications and `LLaDA2.0-flash` (100B parameters) for high-performance scenarios. Both variants retain the parallel decoding advantages of our diffusion training while being optimized for instruction following and safety through comprehensive post-training alignment.

Our contributions provide a practical recipe for the community to leverage AR stability while achieving diffusion parallelism, opening new possibilities for efficient large-scale language modeling.

2 Related Work

2.1 Train dLLMs from scratch

Auto-regressive language models (Ling et al., 2025; Moonshot, 2025; Liu et al., 2024; Meta-AI, 2025) are typically trained by maximizing the likelihood of predicting the next token. Under this paradigm, model performance has been shown to scale effectively with increasing model size, dataset volume, and computational resources, following well-established scaling laws. Recently, MDLMs (Song et al., 2025; Ye et al., 2025; Nie et al., 2025) have emerged as an alternative generative framework, reformulating text generation as an iterative denoising process. In each forward step, a subset of tokens is randomly masked, and the model is trained to recover the original tokens conditioned on the remaining unmasked context.

Encouraged by this paradigm shift, several studies have explored training MDLMs from scratch to assess their full potential. For instance, LLaDA (Nie et al., 2025) demonstrated that a 8B dense MDLM, trained entirely from scratch, achieves performance competitive with similarly sized AR counterparts. Building upon this, LLaDA-MoE (Zhu et al., 2025) introduced the Mixture-of-Experts (MoE) architecture into the MDLM for the first time, showing that a scratch-trained MoE-based MDLM can surpass dense models in both efficiency and capability, thereby validating the compatibility and scalability of MDLMs with advanced MoE designs. Moreover, due to the fundamentally different training dynamics compared to AR models, established training practices and hyperparameter recipes from the AR domain are often suboptimal for MDLMs. To address this gap, recent efforts such as Quakka (Ni et al., 2025) and OpenMoE2 (Ni & team, 2025) have begun investigating the scaling properties and optimal training strategies specifically tailored for MDLMs, laying the groundwork for principled scaling in this emerging paradigm.

However, from-scratch trained MDLMs still lag behind state-of-the-art AR models in overall performance. This gap can be largely attributed to the disparity in training data volume and the maturity of infrastructure support—factors that have been extensively optimized over years of development for AR models. Moreover, due to the high computational cost and long training cycles required for pretraining from scratch, MDLMs mentioned above are typically limited in model scale ($\leq$8B), whereas leading AR models now routinely scale into tens or even hundreds of billions.

2.2 Scaling dLLMs with AR initialization

Given the strong knowledge capacity and performance of AR models, several recent studies have explored initializing dLLMs from pre-trained AR models to reduce training costs and narrow the performance gap between AR models and dLLMs. For instance, DiffusionLLaMA (Gong et al., 2025) and Dream-7B (Ye et al., 2025) adopt a mask annealing strategy to gradually transition from causal attention to bidirectional attention during training, while employing a CART-based loss reweighting scheme to balance token-level learning dynamics. In contrast, RND1 (Keshigeyan et al., 2025) takes a more direct approach by immediately converting the causal attention mechanism of the AR model into a bidirectional one upon initialization. Notably, RND1 observes that when initializing DLM training from an AR model, preserving knowledge-intensive capabilities requires constraining updates to the model's dense layers to prevent catastrophic forgetting.

Block Diffusion Language Models (BDLMs) (Arriola et al., 2025) provide a hybrid paradigm that balances efficiency and performance by combining diffusion and AR modeling. Tokens are generated block-wise: within each block, a diffusion process reconstructs masked tokens, while blocks are produced auto-regressively. This design enables variable-length generation and supports KV-cache reuse during decoding, enhancing inference efficiency. Consequently, BDLMs can be effectively initialized from AR models, narrowing the performance gap. For example, SDAR (Cheng et al., 2025) leverages the Qwen-3 series (Yang et al., 2025) to train more efficient BDLMs. By exploring various block sizes and optimization strategies, it achieves performance comparable to its AR base model.

However, one key limitation across all existing methods is their restricted model scale—ranging only from 7B to 30B parameters—leaving the feasibility and scalability of AR-initialized diffusion models largely unexplored at larger scales. Besides, the low training efficiency of block diffusion hinders its widely application to large-scale corpus for large-size models. Whether such initialization strategies can effectively generalize to models beyond the 30B scale remains an open question.

2.3 dLLMs post-training

Beyond pre-training, post-training is crucial for unlocking the full potential of dLLMs by aligning them with specific tasks and human preferences. This process typically involves supervised fine-tuning (SFT) to instill

instruction-following capabilities, reinforcement learning (RL) to enhance complex reasoning, and inference optimization to address efficiency bottlenecks.

Recent work has explored SFT to adapt dLLMs for specialized domains. For instance, Dream-Coder (Xie et al., 2025) fine-tunes a 7B dLLM for code generation, demonstrating unique abilities like adaptive "sketch-then-fill" strategies for complex algorithms. Similarly, the general-purpose model Dream-7B (Ye et al., 2025) leverages SFT to achieve performance on par with top-tier AR models, while uniquely excelling at tasks requiring complex planning and constraint satisfaction. Other studies have investigated specialized fine-tuning strategies to balance quality and efficiency. Seed-Diffusion (Song et al., 2025), for example, employs a two-stage curriculum learning strategy to train a high-speed code generation model, while LiDAR (Liu et al., 2025) introduces a hybrid "think in diffusion, generate in AR" architecture through fine-tuning, significantly boosting inference throughput while maintaining quality.

To further enhance dLLMs' reasoning abilities, researchers have begun adapting reinforcement learning techniques. However, applying standard policy gradient methods is challenging due to the intractable log-likelihood of dLLMs. To address this, SPG (Wang et al., 2025a) proposes a novel Sandwich Policy Gradient algorithm that obtains a more robust and less biased gradient by maximizing an evidence lower bound for high-reward samples and minimizing an evidence upper bound for low-reward ones. Another line of work, TraceRL (Wang et al., 2025d), focuses on aligning the training objective with the model's multi-step generation trajectory. This framework led to the TraDo series of models, which have not only surpassed strong AR models on reasoning benchmarks but also produced the first dLLM capable of long-chain-of-thought reasoning.

A significant challenge for dLLMs is their slow inference speed, stemming from the iterative nature of the denoising process. To mitigate this, several acceleration methods have been proposed. DPad (Chen et al., 2025a) offers a training-free solution by treating future tokens as a dynamic "scratchpad" and using a sliding window and distance-based pruning to reduce redundant computations, achieving a dramatic speedup, especially for long sequence generation. In contrast, D2F (Wang et al., 2025c) introduces a hybrid autoregressive-diffusion paradigm that enables parallel denoising of future text blocks even before preceding ones are fully generated. This approach allows dLLMs to leverage KV-caching and, for the first time, surpass the inference speed of equivalently sized AR models.

Despite these advances, the field of dLLM post-training is still nascent. Systematic exploration of how these techniques—SFT, RL, and acceleration—interact with one another, and how they scale to models with hundreds of billions of parameters, remains an open and critical area for future research.

3 LLaDA2.0 Training Paradigm

Figure (2) illustrates the holistic training pipeline of LLaDA2.0, a staged and scalable framework designed to transform AR language models into highly efficient diffusion language models. Our paradigm follows a three-stage progression: (1) *Continual Pre-training* from AR to MDLM, (2) *Block Diffusion Pre-training* to transition from token-level to block-level diffusion modeling, and (3) *Post-training* for alignment and task specialization.

The process begins with a strong AR base model. We first perform continual pre-training to adapt this model into an MDLM, where it learns to reconstruct randomly masked tokens in a bidirectional, denoising fashion. This phase bridges the gap between AR and diffusion-based generation while preserving the representational geometry of the original model.

Building upon the trained MDLM, we then introduce *block diffusion pre-training*, during which the model is further trained to denoise contiguous spans of text—referred to as "blocks"—rather than individual tokens. This shift enables higher computational efficiency and better long-range coherence during generation.

Finally, after mastering non-autoregressive generation at both token and block levels, the model undergoes post-training–including SFT and DPO to align its outputs with human intent, instruction-following capability, and downstream application requirements. This stage ensures that the powerful generative backbone developed during diffusion pre-training translates into practical performance gains across diverse tasks.

Overall, LLaDA2.0's training paradigm emphasizes **knowledge inheritance**, **progressive adaptation**, and **efficiency-aware design**, enabling seamless evolution from AR models to fluent, flexible, and fast diffusion large language models.

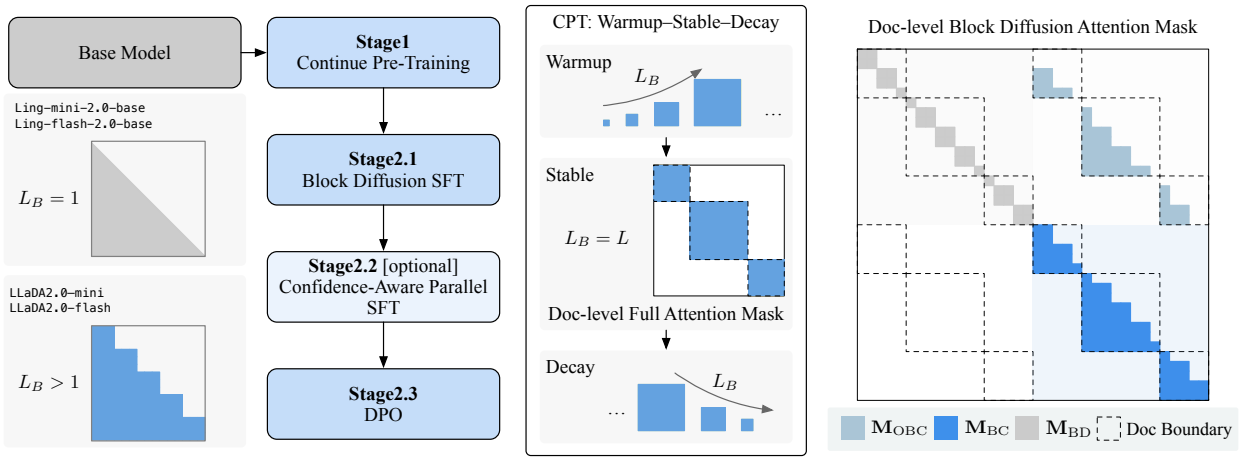

Figure 2: **A schematic of the progressive training framework for transforming an AR model into a MDLM.** Continual Pre-Training Stage facilitates the **Warmup-Stable-Decay** strategies by scheduling block size L_B enables smooth, stable, and effective attention mask adaptation. Post-training Stage facilitates the same block diffusion configuration conducting the instruction SFT, Confidence-Aware Parallel SFT, and DPO. The right panel illustrates the document-level block diffusion attention mask,which enables an efficient, vectorized forward pass by constructing a single input sequence from multiple noisy and clean examples, such as $[\boldsymbol{x}_{\text{noisy1}}, \ldots, \boldsymbol{x}_{\text{clean1}}, \ldots]$. The forward pass then employs a combination of block-diagonal ($\mathbf{M}_{\text{BD}}$), offset block-causal ($\mathbf{M}_{\text{OBC}}$), and block-causal ($\mathbf{M}_{\text{BC}}$) masks.

4 Continual Pre-training via Warmup-Stable-Decay (WSD)

Takeaway

(1) **Warmup-Stable-Decay** enables a smooth and data-efficient conversion from AR to dLLMs.

(2) The **document-level attention mask** ensures coherent bidirectional modeling within semantic boundaries.

(3) **Top-k Checkpoint Merge** enhances performance and generalization by averaging the top k model checkpoints.

Converting a pre-trained AR language model into a high-performance diffusion language model is fundamentally challenging due to the misalignment in architectural inductive biases and training objectives. While AR models generate tokens sequentially from left to right, diffusion-based models rely on bidirectional context and learn to reconstruct corrupted sequences in arbitrary unmasking orders. A direct objective switch often leads to unstable optimization and severe degradation of pretrained knowledge.

To address this gap, we propose a **Warmup–Stable–Decay** (WSD) continual pre-training strategy that enables a smooth, stable, and effective transition from AR to dLLM. WSD decomposes the conversion into three coordinated phases:

- **Warmup**: Progressively increase the block size in block diffusion language models (BDLM) to gradually transform the AR model into a full-sequence masked diffusion language model (MDLM).
- **Stable**: Stabilize and enrich the model's understanding of diffusion dynamics through large-scale training under the MDLM paradigm.
- **Decay**: Revert back to a compact BDLM with smaller block sizes to achieve better speed-efficiency trade-offs during inference.

This progressive schedule preserves the AR model's priors while steadily adapting it to the structural requirements of diffusion modeling.

Moreover, the **document-level attention mask** is applied throughout training to all input sequences. This mechanism is crucial for handling packed heterogeneous documents, preventing the model from forming spurious connections across unrelated texts, thereby ensuring semantic coherence and improving learning

stability within each document. In addition, we adopt a **top-k checkpoint merging** strategy (Tian et al., 2025), to enhance generalization by averaging the parameters of the best-performing checkpoints, smoothing the parameter landscape, and yielding a more robust final model with boosted performance.

4.1 Warmup-Stable-Decay Conversion Strategy

We begin with the AR base models `Ling-mini-2.0` and `Ling-flash-2.0` (Ling et al., 2025), which can be viewed as a special case of BDLM with block size 1. This perspective allows us to treat AR models as the initial BDLM configuration with minimal granularity.

Phase-1: Progressive Block Size Warmup The core idea of the warmup phase is to *gradually increase the block size*, thereby expanding the receptive field within which the model performs joint denoising. Starting from block size $L_B = 1$, we incrementally scale it up to 4, 32, then 64, and ultimately reach $L_B = 4096$ — at which point the entire sequence is treated as one single block. To avoid fragmented blocks, we require the sequence length to be divisible by the current block size. At the final enlargement, the BDLM becomes equivalent to a standard MDLM that operates over fully masked sequences with global attention. Crucially, each block-size transition is trained on moderate-scale data to ensure smooth adaptation. This progressive enlargement allows the model to smoothly adapt its internal representations to handle larger contextual spans and more complex masking patterns.

Phase-2: Large Scale Stable Training Once the block size reaches 4096 and the model transitions to the MDLM pattern, the "clean" part of the attention computation (see Figure 2) no longer needs to be maintained. This significantly reduces the computational cost of attention, allowing data to be processed far more efficiently under the MDLM paradigm. With the model now fully adapted to this regime, the stable training phase focuses on deepening its understanding of diffusion dynamics through extensive training on large-scale corpora. At this stage, the block size is fixed at 4096, effectively making every input a single-block sequence, equivalent to the classical MDLM setting.

Phase-3: Block Size Decay Finally, after large-scale MDLM training, we gradually reduce the block size from 4096 to a small block size (e.g., 32) to convert the model back into an efficient BDLM. This decay process distills the global contextual knowledge learned during MDLM into a compact blockwise structure. By decreasing the block size step-by-step (e.g., starting from 4096 to 2048) rather than abruptly, the model smoothly adapts from global to local conditioning, preserving its semantic understanding while regaining BDLM's practical benefits such as KV-cache reuse and fast variable-length generation.

Overall Training Objective The optimization objective of BDLM (Arriola et al., 2025) is designed to enable the model to accurately reconstruct the original, uncorrupted tokens within these designated masked blocks using a standard cross-entropy loss. Specifically, we define the training loss during warmup and decay phases (phase-1&3) under the BDLM paradigm as:

$$\mathcal{L}_{\text{BDLM}}(\theta) = -\mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_t}\left[\frac{\alpha_t'}{1-\alpha_t}\sum_{k=1}^{K}\sum_{i=1}^{L_B}\mathbb{1}[x_{t,k}^i = [\text{MASK}]]\log p_\theta(\boldsymbol{x}_{0,k}^i|\boldsymbol{x}_{0,<k},\boldsymbol{x}_{t,k})\right], \tag{1}$$

where the expectation is over timestep t, the clean sequence $\boldsymbol{x}_0$, and its corrupted version $\boldsymbol{x}_t$ (tokens masked with probability $1-\alpha_t$). Indicator $\mathbb{1}[\cdot]$ ensures predictions are made only for masked tokens, and $-\alpha_t'/(1-\alpha_t)$ is the diffusion-derived time weight. Here $K = L_{\text{total}}/L_B$ is the number of blocks, L_B the block size, $x_{t,k}^i$ the i-th token in block k, $\boldsymbol{x}_{0,<k}$ the preceding clean blocks, and $\boldsymbol{x}_{t,k}$ the noisy version of the current block.

During the stable training (phase-2) of MDLM (i.e., K=1), the objective simplifies to:

$$\mathcal{L}_{\text{MDLM}}(\theta) = -\mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_t}[\frac{\alpha_t'}{1-\alpha_t}\sum_{i=1}^{L}\mathbb{1}[x_t^i = [\text{MASK}]]\log p_\theta(x_0^i|\boldsymbol{x}_t)]. \tag{2}$$

4.2 Document-level Attention Mask

Our training sequences are formed by packing heterogeneous documents into fixed-length segments to maximize throughput. However, this introduces artificial long-range dependencies across semantically unrelated texts. Without careful handling, standard attention would incorrectly attend across document boundaries, leading to contextual confusion and significantly hindering the model's ability to perform robust bidirectional modeling crucial for denoising.

To mitigate this fundamental challenge and preserve semantic coherence, we redefine the attention mechanism with a specialized **block-wise document-level attention mask**. This mask ensures that attention operates

strictly within document boundaries, preventing cross-document contamination and allowing the model to fully leverage bidirectional context for accurate reconstruction of corrupted blocks. The native Block Diffusion vectorizes the training process to achieve parallel training of blocks, and this mask is applied accordingly. Specifically, for a concatenated sequence x_{full} of length $2L$ (comprising x_t followed by x_0), and assuming tokens i and j are already confined to the same document segment (as enforced by the initial document-level mask), the attention mask $\boldsymbol{M} \in \{0,1\}^{2L\times 2L}$ is constructed by dividing each sequence (x_t and x_0) into contiguous blocks. Let $b(k) = \lfloor k/L_B \rfloor$ denote the block index for token k given a block size L_B. The mask is defined as:

$$\boldsymbol{M}_{ij} = \begin{cases} \mathbb{1}_{b(i)=b(j)} & \text{if } i \in x_t \text{ and } j \in x_t \\ \mathbb{1}_{b(i)>b(j-L)} & \text{if } i \in x_t \text{ and } j \in x_0 \\ \mathbb{1}_{b(i-L)\geq b(j-L)} & \text{if } i \in x_0 \text{ and } j \in x_0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where $i, j \in \{0, 1, \ldots, 2L-1\}$ are the indices in the full sequence. The first condition ($\mathbb{1}_{b(i)=b(j)}$) implements block-diagonal attention within the noisy sequence x_t. The second ($\mathbb{1}_{b(i)>b(j-L)}$) enables cross-attention from x_t to x_0, but only from blocks in x_t to earlier blocks in x_0. The third ($\mathbb{1}_{b(i-L)\geq b(j-L)}$) imposes a causal block attention pattern within the clean sequence x, allowing a block to attend to itself and all preceding blocks. The "otherwise" condition corresponds to a zero matrix, explicitly preventing attention from queries in x_0 to keys in x_t. This allows each block to leverage context from relevant blocks (according to the mask) for reconstruction, capturing inter-block dependencies while maintaining the causal and block-diagonal principles essential for stable diffusion training. During our exploration, we also experimented with other tricks like random-length (Xie et al., 2025) and CART (Ye et al., 2025). However, the results demonstrate that the document-level attention mask is more fundamental in CPT training compared to these techniques, and it consistently achieves superior performance. As illustrated in Figure 2, this forms a structured attention layout that balances locality and global document coherence.

For MDLM, the document-level attention mask simplifies to $\boldsymbol{M} \in \{0,1\}^{L\times L}$, where:

$$\boldsymbol{M}_{ij} = \begin{cases} 1, & \text{if } i, j \text{ belong to the same document,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

4.3 Top-k Checkpoint Merge

To further enhance the generalization and robustness of our Block Diffusion Language Model, we employ a top-k checkpoint merging strategy. Upon completion of BDLM pre-training, we identify the top k best-performing model checkpoints, typically selected based on validation metrics like perplexity. The parameters (weights and biases) of these k checkpoints are then arithmetically averaged to form a single, unified BDLM. Based on WSM scheduler (Tian et al., 2025), this merge strategy can effectively ensemble diverse "knowledge" captured by the model at various optimal or near-optimal training states. This smooths the parameter landscape, mitigates overfitting, and yields a more stable and generalizable model. A key advantage of the WSM approach is its optimizer-agnostic nature, allowing seamless integration without altering the underlying training pipeline. Crucially, this post-training Top-k Merge fundamentally differs from the Exponential Moving Average (EMA). While EMA is an in-training technique that continuously smooths parameters, merging is an offline procedure. It explicitly selects and averages distinct, high-performing model states, consolidating their strengths rather than merely smoothing the final training step.

5 Post-training

Takeaway

(1) Applying **complementary masking** and a **mask ratio bandwidth** during SFT improves sample efficiency and stabilizes convergence.

(2) An auxiliary **confidence loss** is incorporated to sharpen predictions, which is crucial for efficient parallel decoding.

(3) **DPO** is adapted by defining sequence log-probabilities over masked tokens, enabling effective preference alignment for the diffusion model.

5.1 Supervised Fine-Tuning with Block Diffusion

Following the pre-training phase, the model is aligned to follow user instructions through supervised fine-tuning (SFT). This is achieved by adapting the diffusion training objective to be conditional on an input prompt, $\boldsymbol{c}$. The model is thus trained to generate the desired response $\boldsymbol{x}_0$ by minimizing the following loss function:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{t,(\boldsymbol{c},\boldsymbol{x}_0),\boldsymbol{x}_t}\left[\frac{\alpha'_t}{1-\alpha_t}\sum_{k=1}^{K}\sum_{i=1}^{L_B}\mathbb{1}[x^i_{t,k} = [\text{MASK}]]\log p_\theta(x^i_{0,k}|\boldsymbol{c}, \boldsymbol{x}_{0,<k}, \boldsymbol{x}_{t,k})\right]. \tag{5}$$

Here, the model p_θ learns to predict the original tokens $x^i_{0,k}$ of a clean response from a noisy version $\boldsymbol{x}_t$. The loss is computed only on masked tokens within the current noisy block $\boldsymbol{x}_{t,k}$. To do this, the prediction is conditioned on the prompt $\boldsymbol{c}$, auto-regressive context from prior clean blocks $\boldsymbol{x}_{0,<k}$, and the current noisy block $\boldsymbol{x}_{t,k}$ that it must denoise.

Padding strategies & Mask ratio bandwidth To ensure compatibility with our block-wise attention mask, we quantize each sequence's length. Specifically, the original length is rounded up to the nearest multiple of the block size, b. This process defines an "effective length" for each sequence, guaranteeing its boundaries align perfectly with the block boundaries required by the attention mechanism.

To optimize the training dynamics, we further implement a "mask ratio bandwidth" strategy. Standard discrete diffusion processes typically sample mask probabilities across the full unit interval, $\alpha_t \sim U[0,1]$. However, as identified by Arriola et al. (2025), extreme masking rates induce high gradient variance while offering minimal learning signal: near-zero masking renders reconstruction trivial, while near-total masking reduces the objective to simply learning data marginals. To mitigate this, we clip the noise schedule, constraining the sampling of mask rates to a bounded interval $[\alpha_{\min}, \alpha_{\max}]$ rather than the full range. This bandwidth restriction focuses the training objective on the noise regimes that provide the most informative gradients, thereby stabilizing convergence and improving the model's generative perplexity.

Complementary Masking Complementary Masking (Li et al., 2025) is a training optimization that enhances the data efficiency of the MDLM objective, $\mathcal{L}_{\text{MDLM}}(\theta)$. The strategy's core principle is to generate two antithetical training instances from a single source sequence $\boldsymbol{x}_0$. A primary noised sequence, $\boldsymbol{x}_t$, is formed using a random mask, while a complementary sequence, $\boldsymbol{x}'_t$, is simultaneously produced using that mask's logical inverse[1].

By incorporating both $\boldsymbol{x}_t$ and $\boldsymbol{x}'_t$ into the same training batch, this method provides a deterministic guarantee: every token position across the sequence length L is presented to the model in its uncorrupted state exactly once within the pair. This not only doubles the effective data utilization from each sample, thereby accelerating convergence, but also entirely eliminates token-level sampling bias. Consequently, the model benefits from a more comprehensive and uniform learning signal at every optimization step, leading to enhanced robustness.

Data Recipe Curation A balanced, high-quality SFT dataset underpins the model's capabilities, achieved through a strategic composition of tasks spanning three principal pillars: Reasoning, General, and Industrial. The Reasoning pillar hones analytical and logical faculties through mathematics and code generation. The General pillar cultivates linguistic richness and social intelligence via creative and dialogic tasks. The Industrial pillar embeds domain-specific expertise by simulating end-to-end workflows under real-world constraints. This integrated methodology ensures a holistic skill profile, preventing capability skew and enabling fluid shifts between abstract reasoning and applied problem-solving.

5.2 Confidence-Aware Parallel Training

To enhance the model's predictive confidence, which is crucial for efficient parallel decoding, we propose Confidence-Aware Parallel (CAP) Training. We incorporate an auxiliary confidence loss, $\mathcal{L}_{\text{conf}}$, inspired by dParallel (Chen et al., 2025b). The primary objective, $\mathcal{L}_{\text{SFT}}$, ensures correctness but provides diminishing incentive to sharpen the predictive distribution for tokens that are already correctly predicted. The confidence loss addresses this by selectively minimizing the entropy of the model's output distribution, $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{c})$, but only for the subset of tokens that are correctly predicted in a given step. This compels the model to increase its certainty on its correct predictions. The final training objective is a weighted combination of the two losses:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{SFT}}(\theta) + \lambda\mathcal{L}_{\text{conf}}(\theta), \tag{6}$$

[1]We also try complementary masking in CPT and find it only works fine on corpus less than 100B tokens, while it does not show advantages with more training data, so that we only adopt it in post-training.

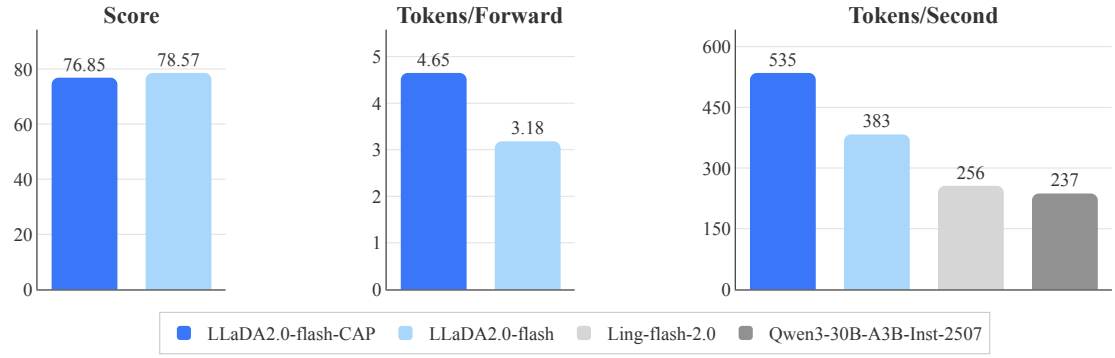

Figure 3: Average score and tokens-per-forward (TPF) for LLaDA2.0-flash with and without CAP across 12 benchmarks. Inference speed (tokens per second) of LLaDA2.0-flash compared with similarly sized AR models on 4 code and math benchmarks.

where λ is a hyperparameter that balances the two objectives. As illustrated in Figure 3, CAP training effectively improves the decoding efficiency of LLaDA2.0-flash while maintaining competitive compression performance, demonstrating a favorable trade-off between generation quality and inference speed.

5.3 DPO

Building upon the SFT stage, we further align the policy model π_θ with human intent using Direct Preference Optimization. To support this, we constructed a comprehensive dataset comprising 1.5 million preference pairs across diverse domains, including general knowledge, mathematics, and instruction following. To ensure a stable transition in optimization, the learning rate for the DPO stage is initialized consistently with the final learning rate of the preceding SFT phase.

Since the policy model π_θ is trained to reconstruct clean tokens $\boldsymbol{x}_0$ from noisy blocks $\boldsymbol{x}_t$ conditioned on context $\boldsymbol{c}$, the standard DPO formulation—which requires exact log-likelihoods—is intractable. Following established practices for diffusion models, we substitute the conditional log-likelihoods with their ELBO. We first define the conditional Block Diffusion ELBO, $B_{\text{BDLM}}(\theta, \boldsymbol{x}|\boldsymbol{c})$, for a response $\boldsymbol{x}$. This term mirrors the inner objective of our SFT loss (equation 5) and is estimated via a single Monte Carlo sample over timesteps and noise:

$$B_{\text{BDLM}}(\theta, \boldsymbol{x}|\boldsymbol{c}) = \mathbb{E}_{t,\boldsymbol{x}_t}\left[\frac{\alpha_t'}{1-\alpha_t}\sum_{k=1}^{K}\sum_{i=1}^{L_B}\mathbb{1}[x_{t,k}^i = [\text{MASK}]]\log p_\theta(x_k^i|\boldsymbol{c}, \boldsymbol{x}_{<k}, \boldsymbol{x}_{t,k})\right]. \tag{7}$$

Given a preference pair $(\boldsymbol{x}_w, \boldsymbol{x}_l)$, where $\boldsymbol{x}_w$ is the preferred response and $\boldsymbol{x}_l$ is the dispreferred response, the DPO objective maximizes the margin between the ELBO estimates of the policy π_θ and the frozen reference model $\pi_{\theta_{\text{ref}}}$ (initialized from the post-SFT model). The final loss function is defined as:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(\boldsymbol{c},\boldsymbol{x}_w,\boldsymbol{x}_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\left[\Delta B(\boldsymbol{x}_w|\boldsymbol{c}) - \Delta B(\boldsymbol{x}_l|\boldsymbol{c})\right]\right)\right], \tag{8}$$

where $\Delta B(\boldsymbol{x}|\boldsymbol{c}) = B_{\text{BDLM}}(\theta, \boldsymbol{x}|\boldsymbol{c}) - B_{\text{BDLM}}(\theta_{\text{ref}}, \boldsymbol{x}|\boldsymbol{c})$ represents the ELBO advantage of the policy over the reference model, and β is a hyperparameter (set to 0.1) that controls the deviation from the reference policy.

5.4 Inference

We sample one block at a diffusion step, conditioned on previously sampled blocks $p_\theta(\boldsymbol{x}_s^b|\boldsymbol{c}, \boldsymbol{x}_t^{<b})$. The generation of each block is itself a multi-step iterative refinement process. At each step, candidate tokens are sampled for all remaining unfilled positions within the block. A hybrid acceptance strategy is then employed: we first accept all tokens whose sampling probability exceeds a predefined confidence 'threshold'. If an insufficient number of tokens meet this criterion, a low-confidence fallback is triggered, where we instead accept a fixed number of the most probable tokens regardless of their absolute confidence. This dual mechanism ensures steady generation progress.

6 Evaluation

6.1 Setup

To comprehensively evaluate the quality of instruction-tuned models, we employ a diverse suite of benchmarks categorized into five dimensions:

- **Knowledge**: MMLU (Hendrycks et al., 2020), MMLU-Pro (Wang et al., 2024), GPQA-Diamond (Rein et al., 2024), ARC (Clark et al., 2018), CMMLU (Li et al., 2023a) C-Eval (Huang et al., 2023), GAOKAO-Bench (Zhang et al., 2023), SciBench (Wang et al., 2023), PHYBench (Qiu et al., 2025), TriviaQA (Joshi et al., 2017)
- **Reasoning**: SQuAD 2.0 (Rajpurkar et al., 2018), DROP (Dua et al., 2019), KOR-Bench (Ma et al., 2024), HellaSwag (Zellers et al., 2019), BIG-Bench Hard (Suzgun et al., 2023), BIG-Bench Extra Hard (Kazemi et al., 2025), MuSR (Sprague et al., 2023), ZebraLogic (Lin et al., 2025), PrOntoQA (Saparov & He, 2022), PIQA (Bisk et al., 2020), OCNLI (Hu et al., 2020), BIG-Bench Hard-CN (team, 2023c)
- **Coding**: CRUXEval (Gu et al., 2024), MBPP (Austin et al., 2021), MultiPL-E (Cassano et al., 2023), HumanEval (Chen et al., 2021), BigCodeBench (Zhuo et al., 2024), LiveCodeBench (Jain et al., 2024), Spider (Yu et al., 2018), BIRD (Li et al., 2023b), HumanEval+ (Liu et al., 2023), MBPP+ (Liu et al., 2023), HumanEvalFix (Muennighoff et al., 2023), Aider (team, 2023a), HumanEval-CN (team, 2023c)
- **Math**: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), AIME 2025 (AIME, 2025), Omni-MATH (Gao et al., 2024), HARDMath2 (Roggeveen et al., 2025), GSM-Plus (Li et al., 2024), CMATH (Wei et al., 2023)
- **Agent & Alignment**: BFCL (Patil et al., 2025), IFEval (Zhou et al., 2023), CodeIF-Bench (Wang et al., 2025b), Nexus Function Calling Benchmark (team, 2023b)

This extensive evaluation suite, comprising a total of 47 benchmarks, provides a holistic foundation for assessing model capabilities. In our experiments, we compare the LLaDA2.0 series against strong open-source auto-regressive (AR) models.

For all LLaDA2.0 models, we utilize a temperature of 0.0, a block size of 32, and a decoding threshold of 0.95.

6.2 Results

The overall results, presented in the following tables, indicate that the LLaDA2.0 architecture is not only highly competitive, but also shows a promising trend of closing the performance gap with, and even surpassing, AR models in specific key areas. Our models consistently demonstrate strong, and often superior, performance in complex, structured tasks. For instance, LLaDA2.0-mini already outperforms a comparable AR model (Qwen3-8B) in the domains of Reasoning, Coding, and Math. This signal is amplified in our larger model, as LLaDA2.0-flash achieves parity with the powerful Qwen3-30B-A3B-Instruct-2507 and establishes a lead in the critical Coding and Agent domains. This suggests that as diffusion models scale, their inherent strengths in structured generation and tool use become increasingly apparent.

As shown in Table 1, **LLaDA2.0-mini** achieves a competitive average score of 64.34, closely approaching its AR peer, Ling-mini-2.0 (65.77). This demonstrates the fundamental viability of the diffusion approach. More importantly, it shows promising signals in complex tasks, outperforming its direct competitor on reasoning benchmarks like SQuAD 2.0 (86.50) and demonstrating more robust instruction following on IFEval (80.78). Its strong performance in coding tasks such as HumanEval (86.59) further suggests an early aptitude for structured generation.

This potential becomes even more evident with our larger model, **LLaDA2.0-flash**. As shown in Table 2, with an average score of 73.18, it stands firmly on par with strong AR models such as Qwen3-30B-A3B-Instruct-2507 (73.60). Crucially, LLaDA2.0-flash begins to exhibit clear advantages in complex generative tasks, a sign that the diffusion architecture may hold inherent strengths. In the critical domain of coding, it consistently outperforms its AR peers, scoring higher on HumanEval (94.51), MBPP (88.29) and MultiPL-E (74.87). This trend of surpassing AR models also extends to agent capabilities (BFCL v3: 75.43) and advanced mathematics (AIME 2025: 60.00).

In conclusion, the LLaDA2.0 series successfully demonstrates that diffusion-based language models are a powerful and scalable alternative to the dominant auto-regressive paradigm. While rapidly narrowing the gap on general benchmarks, they are already showcasing the potential to surpass traditional architectures in complex, structured domains like code generation and tool use. This positions diffusion models as a highly promising direction for the future of language generation.

Table 1: Benchmark Performance of LLaDA2.0-mini

Benchmark	Qwen3-8B (no_think)	Ling-mini-2.0	LLaDA2.0-mini-preview	LLaDA2.0-mini
Average	63.42	65.77	54.67	64.34
		Knowledge		
MMLU	80.94	82.15	72.49	80.53
MMLU-Pro	65.48	63.72	49.22	63.22
CMMLU	79.17	80.84	67.53	79.50
C-EVAL	81.36	82.10	66.54	81.38
GAOKAO-Bench	84.94	87.23	74.46	84.30
ARC-c	93.35	93.09	89.15	93.56
GPQA	46.59	56.80	23.74	47.98
SciBench	2.85	5.28	4.10	3.53
PHYBench	9.76	14.59	5.08	11.70
TriviaQA	52.51	55.63	50.49	51.33
		Reasoning		
BIG-Bench Hard	79.48	83.70	70.64	78.21
BIG-Bench Extra Hard	18.27	14.81	12.36	16.47
bbh-zh	80.09	66.11	66.62	75.75
MuSR	70.02	71.36	56.77	71.48
ZebraLogic	37.48	79.85	14.80	64.20
PrOntoQA	93.12	96.06	70.00	86.00
PIQA	88.30	87.54	84.33	86.51
OCNLI	61.49	60.17	58.68	64.51
HellaSwag	79.56	69.02	74.01	79.01
KOR-Bench	54.48	62.72	37.26	50.40
DROP	84.56	78.80	79.49	81.91
SQuAD 2.0	85.21	75.56	85.61	86.50
		Coding		
CRUXEval-O	74.06	76.12	61.88	71.62
MBPP	78.92	84.07	77.75	81.50
MBPP+	71.96	76.46	66.67	74.07
MultiPL-E	61.70	67.09	62.43	67.46
HumanEval	84.76	85.98	80.49	86.59
HumanEval+	78.66	81.71	71.95	79.88
HumanEvalFix	76.02	82.83	60.16	74.90
HumanEval-cn	74.39	71.34	73.17	78.66
BigCodeBench-Full	36.05	35.00	30.44	32.89
LiveCodeBench	26.38	34.97	19.82	31.50
Aider	55.64	49.62	28.57	39.85
BIRD-SQL	36.11	39.67	27.71	39.34
Spider	72.80	76.43	75.64	76.76
		Math		
GSM8K	93.63	94.62	89.01	94.24
MATH	86.28	94.66	73.50	93.22
OlympiadBench	55.33	72.30	36.30	67.70
AIME 2025	22.08	47.66	10.00	36.67
HARDMath2	7.58	9.95	0.95	0.47
Omni-MATH	33.20	48.80	19.20	41.70
GSM-Plus	86.09	87.82	81.44	86.24
CMATH	95.42	96.40	90.53	95.72
		Agent & Alignment		
IFEval-strict-prompt	86.90	76.16	62.50	80.78
BFCL v3	70.08	53.98	74.11	70.90
CodeIF-Bench	50.00	46.00	48.00	48.00
Nexus FC	37.71	34.38	33.68	35.18

Table 2: Benchmark Performance of LLaDA2.0-flash

Benchmark	Qwen3-30B-A3B-Instruct-2507	Ling-flash-2.0	LLaDA2.0-flash-preview	LLaDA2.0-flash
Average	73.60	72.15	65.97	73.18
		Knowledge		
MMLU	87.13	87.98	83.15	87.69
MMLU-Pro	74.23	76.84	66.16	73.36
CMMLU	86.36	86.59	79.64	85.13
C-EVAL	88.17	88.03	79.28	86.75
GAOKAO-Bench	94.53	93.24	86.12	93.90
ARC-c	95.81	95.08	93.90	95.93
GPQA	57.34	67.12	41.92	61.98
SciBench	4.54	4.14	5.13	4.13
PHYBench	29.84	27.67	7.58	30.06
TriviaQA	65.61	69.76	69.25	66.88
		Reasoning		
BIG-Bench Hard	85.54	89.36	82.85	86.75
BIG-Bench Extra Hard	37.80	23.24	16.70	27.86
BIG-Bench Hard - CN	86.18	75.09	83.38	87.52
MuSR	79.15	82.72	78.75	80.48
ZebraLogic	90.97	87.60	39.90	82.30
PrOntoQA	97.12	97.88	93.50	96.50
PIQA	91.57	91.95	91.84	92.76
OCNLI	71.59	65.36	69.39	71.63
HellaSwag	86.31	81.59	86.00	84.97
KOR-Bench	68.00	68.96	53.28	64.24
DROP	87.57	88.32	88.17	87.90
SQuAD 2.0	89.51	81.32	90.61	90.00
		Coding		
CRUXEval-O	86.75	82.75	74.50	85.12
MBPP	86.65	85.01	86.65	88.29
MBPP+	78.04	76.19	75.93	79.63
MultiPL-E	70.67	65.76	72.38	74.87
HumanEval	93.29	85.98	88.41	94.51
HumanEval+	88.41	85.98	82.32	87.80
HumanEvalFix	91.16	92.68	83.33	90.24
HumanEval-CN	87.20	74.39	84.76	89.02
Bigcodebench-Full	41.49	40.70	40.44	41.58
LiveCodeBench	41.63	44.11	29.07	42.29
Aider	71.43	71.43	51.13	66.92
Spider	81.79	80.58	81.37	82.49
BIRD-SQL	47.75	47.49	45.34	45.76
		Math		
GSM8K	96.36	95.45	95.75	96.06
MATH	96.70	96.10	83.52	95.44
OlympiadBench	77.59	76.19	49.33	74.07
AIME 2025	61.88	55.89	23.33	60.00
HARDMath2	4.27	23.70	3.79	4.27
Omni-MATH	54.00	53.00	24.60	50.30
GSM-Plus	89.45	89.83	88.25	89.64
CMATH	96.58	96.52	95.26	96.90
		Agent & Alignment		
IFEval-strict -prompt	84.29	81.52	75.60	81.70
BFCL v3	73.19	67.57	74.86	75.43
CodeIF-Bench	54.00	56.00	56.00	58.00
Nexus FC	49.93	36.25	47.98	50.45

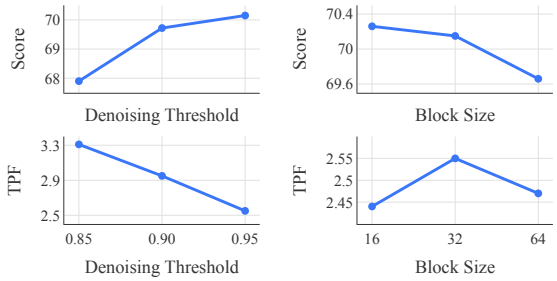

Figure 4: Score/TPF vs threshold/block size

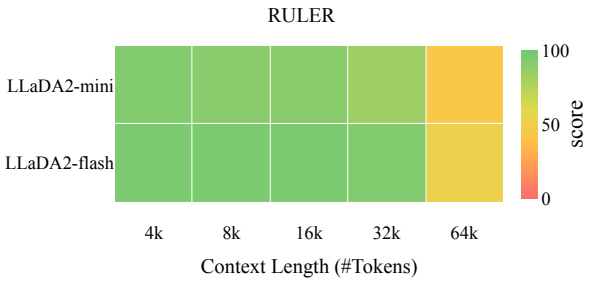

Figure 5: Performance on the RULER benchmark.

6.3 Analysis

Analysis of Inference Hyper-parameters In addition to our main evaluation, we conducted a brief analysis to tune key inference hyperparameters. To ensure efficiency, this analysis was performed on our LLaDA2.0-mini model, using a representative subset of our benchmarks to understand the trade-off between generation quality (score) and inference speed (measured as TPF - Tokens Per Forward; higher is faster).

Denoising Threshold. We first investigate the impact of the Denoising Threshold. While keeping the Block Size fixed at 32, we varied the threshold and observed its effect on quality and speed. As shown in Figure 4, the results reveal a clear trade-off. A threshold of 0.95 achieved the highest quality score (70.15) at the cost of the lowest inference speed (2.55 TPF). Lowering the threshold to 0.85 boosted the speed to its peak (3.31 TPF), but led to an unacceptable degradation in quality, with the score dropping to 67.90.

Block Size. Subsequently, we analyze the effect of Block Size. We set the Denoising Threshold to 0.95, the optimal value identified in the prior experiment. The results in Figure 4 demonstrate a similar trade-off. A block size of 16 yielded the highest score (70.26) but with the slowest inference (2.44 TPF). In contrast, increasing the block size to 32 substantially improved the speed to 2.55 TPF with only a marginal quality drop to 70.15. Further increasing the block size to 64 proved suboptimal, as it degraded both score and speed relative to the size-32 setting. Therefore, a block size of 32 emerges as the most compelling choice, offering a significant speed-up for a negligible performance cost.

In summary, based on this analysis, the configuration for our main evaluation is well-supported. The Denoising Threshold of 0.95 is the clear choice for maximizing quality. For block-size, the setting of 32 represents an optimal balance, providing the highest throughput with virtually no sacrifice in performance compared to the slightly higher-scoring but slower setting of 16.

Analysis of Context Length To rigorously validate our model's performance across various context lengths, we conducted a series of evaluations using the RULER benchmark.

As shown in Figure 5, both models demonstrate strong performance and stability within context length of 32k. The LLaDA2.0-flash model is particularly robust, maintaining a score above 93 across all lengths from 4k to 32k. The LLaDA2.0-mini model also achieves high scores, starting at 93.29 for 4k but showing a degradation to 83.94 at 32k.

To test the models' extrapolation capabilities, we extended the context length to 64k. This was achieved by employing dynamic RoPE scaling during inference, specifically using the YaRN method with a scaling factor of 2.0. However, this extension resulted in a performance degradation for both models, demonstrating a clear trade-off between context length extension and task accuracy.

In summary, this evaluation highlights two key findings: (1) The LLaDA2.0 models are exceptionally robust for long-context tasks within their native 32k window. (2) They can be successfully extended to handle 64k sequences via YaRN scaling, providing flexibility for extreme-length applications, albeit with a predictable performance cost.

7 Training & Inference Infrastructure

7.1 Pretraining

We adopt Megatron-LM (Shoeybi et al., 2019) as the pretraining backend to enable efficient training of a 100B-parameter model with long sequences, leveraging data parallelism (DP), pipeline parallelism (PP),

tensor parallelism (TP), context parallelism (CP), and expert parallelism (EP), as Figure 6 shows. To ensure consistency of masked tokens, we generate masked tokens on a single model-parallel (MP, that is, TP and PP) rank and then broadcast to all other ranks within the MP ranks.

Efficient Block Diffusion Training For flexible support of arbitrary block diffusion attention mask, we utilize cuDNN as the backend for the attention mechanism. This approach achieves more than 1.3x end-to-end speedup and over 90% memory savings in the attention layer compared to the unfused attention implementation in TransformerEngine when training LLaDA2.0-mini. We further apply a zig-zag partitioning strategy to the block diffusion attention mask to achieve effective load balancing across the CP group.

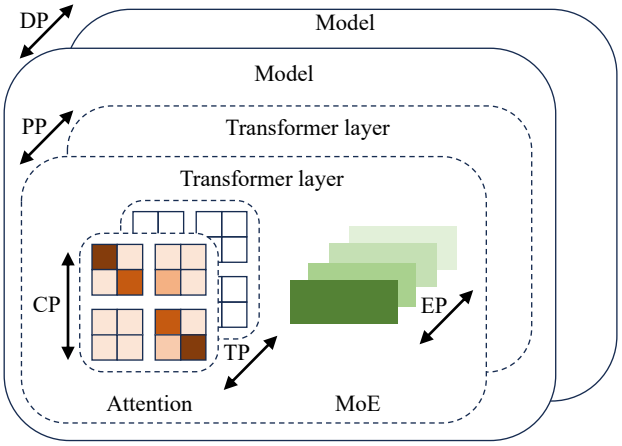

Figure 6: Parallelism overview.

Numerical Stability During the transition from AR to diffusion models, training can suffer from gradient explosion, especially at high mask ratios within a document. This issue stems from the fact that masked token embeddings are set to zero during AR training, as these tokens are never observed, leading their corresponding weights to gradually decay to zero. A straightforward fix, randomly reinitializing the masked token embeddings upon loading the AR model, may disrupt other well-trained parameters, potentially causing catastrophic forgetting. To mitigate this while preserving pre-trained knowledge, we instead add independent Gaussian noise to the output of the embedding layer for each masked token during the initial iterations of training. This ensures that the L2 norm of the masked token's embedding remains significant to avoid gradient explosion, thereby stabilizing the training process.

7.2 Post-Training

For the post-training phase, we leverage dFactory[2] (InclusionAI, 2025), a repository providing efficient training recipes for dLLMs. Built upon the VeOmni (Ma et al., 2025a) distributed training framework, dFactory allows us to effectively implement complex parallelization schemes. Specifically, our setup for fine-tuning LLaDA2.0 combines Data Parallelism (DP) and Expert Parallelism (EP) to ensure scalable and stable training. To further enhance data throughput and hardware utilization, we adopt a data packing strategy analogous to those used in continued pre-training, which concatenates multiple short sequences into a single longer sequence. This integrated approach provides a robust and high-performance infrastructure for the post-training of our model.

7.3 Inference Engine

We adapt dInfer[3] (Ma et al., 2025b)–originally built for high-performance diffusion LLM inference–to efficiently support block diffusion inference. This requires the inference engine to leverage optimization techniques traditionally designed for AR models. For instance, the framework can now effectively exploit KV-cache reuse to substantially reduce prefill computation. As block diffusion inference closely resembles auto-regressive generation in the execution pattern, we also incorporated block diffusion inference support into SGLang[4] (Zheng et al., 2024), allowing it to benefit from the same class of system-level optimizations designed for AR models. More mature features in dInfer are undergoing to transport to SGLang.

Inference speed Figure 3 compares the average inference throughput (Tokens Per Second, TPS = #decoding tokens/#total-time) of our optimized LLaDA2.0-flash models against state-of-the-art AR models of similar scale on four reasoning and code-generation benchmarks (HumanEval, MBPP, GSM8K, and CRUXEval). All models are evaluated under a consistent generation setup. For diffusion-based models (LLaDA2.0-flash and LLaDA2.0-flash-CAP), we adopt a threshold decoder with a threshold of 0.95. The AR baselines (Ling-flash-2.0 and Qwen3-30B-A3B-Instruct-2507) are deployed using SGLang, while the diffusion models

[2]https://github.com/inclusionAI/dFactory
[3]https://github.com/inclusionAI/dInfer
[4]https://github.com/sgl-project/sglang/issues/12766

are served with dInfer, ensuring fair performance comparison in real inference environments. As shown, LLaDA2.0-flash-CAP reaches 535 TPS, outperforming the standard LLaDA2.0-flash (383 TPS) and providing up to 2.1× speed-up over the AR baselines (256 TPS and 237 TPS).

8 Conclusion

In this work, we introduced LLaDA2.0, discrete diffusion language models scaling up to 100B total parameters through systematic conversion from auto-regressive models, as well as a set of novel and comprehensive recipes designed to smooth and effectively transform traditional AR language models into highly efficient and performant Masked Diffusion Language Models.

Through extensive evaluations, it validates the feasibility of the training paradigm. The LLaDA2.0-mini and LLaDA2.0-flash models achieve performances that are competitive with their AR counterparts. Slightly surprisingly, LLaDA2.0-flash seems to have demonstrated advantages in complex, structured domains such as code generation, mathematical reasoning, and agentic tool use. These may have opened a new door to future work in the agentic LLM era while solidifying a gaugeable potential of dLLM for test-time scaling.

Future work may point to further scaling of the parameter volume, RL/thinking paradigm and extending the decoding speed to its extreme.

References

AIME. AIME Problems and Solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

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:2503.09573*, 2025.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691, 2023.

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

Xinhua Chen, Sitao Huang, Cong Guo, Chiyue Wei, Yintao He, Jianyi Zhang, Hai "Helen" Li, and Yiran Chen. DPad: Efficient Diffusion Language Models with Suffix Dropout, August 2025a. arXiv:2508.14148.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable Parallel Decoding for dLLMs. *arXiv:2509.26488*, 2025b.

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: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:1803.05457*, 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:2110.14168*, 2021.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning over Paragraphs. *arXiv:1903.00161*, 2019.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv:2410.07985*, 2024.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv:2407.21783*, 2024.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. CruxEval: A Benchmark for Code Reasoning, Understanding and Execution. *arXiv:2401.03065*, 2024.

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:2402.14008*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving with the Math Dataset. *arXiv:2103.03874*, 2021.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence S Moss. Ocnli: Original chinese natural language inference. *arXiv:2010.05444*, 2020.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models. *Advances in Neural Information Processing Systems*, 36:62991–63010, 2023.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o System Card. *arXiv:2410.21276*, 2024.

InclusionAI. dFactory: Easy and Efficient dLLM Fine-Tuning, 2025. URL https://github.com/inclusionAI/dFactory.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv:2403.07974*, 2024.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv:1705.03551*, 2017.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025.

Chandrasegaran Keshigeyan, Thomas Armin, and others at Radical Numerics. Training diffusion language models at scale using autoregressive models. https://github.com/RadicalNumerics/RND1, 2025.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring Massive Multitask Language Understanding in Chinese. *arXiv:2306.09212*, 2023a.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357, 2023b.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv:2402.19255*, 2024.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. LaViDa: A Large Diffusion Language Model for Multimodal Understanding, June 2025. arXiv:2505.16839.

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

Team Ling, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, Chao Zhang, et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation. *arXiv:2510.22115*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 Technical Report. *arXiv:2412.19437*, 2024.

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

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in Diffusion, Talk in Autoregression, November 2025. arXiv:2511.08923.

Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. *arXiv:2410.06526*, 2024.

Qianli Ma, Yaowei Zheng, Zhelun Shi, Zhongkai Zhao, Bin Jia, Ziyue Huang, Zhiqi Lin, Youjie Li, Jiacheng Yang, Yanghua Peng, et al. Veomni: Scaling any modality model training with model-centric distributed recipe zoo. *arXiv:2508.02317*, 2025a.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qiand Xinyuan Zhang, et al. dinfer: An efficient inference framework for diffusion language models. *arXiv:2510.08666*, 2025b.

Meta-AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Moonshot. Kimi K2. https://github.com/MoonshotAI/Kimi-K2/, 2025.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. In *NeurIPS 2023 workshop on instruction tuning and instruction following*, 2023.

Jinjie Ni and team. Openmoe 2: Sparse diffusion language models. https://jinjieni.notion.site/OpenMoE-2-Sparse-Diffusion-Language-Models-277d8f03a8668065a4ecd23f23bd6aac, 2025. Notion Blog.

Jinjie Ni, Qian Liu, Chao Du, Longxu Dou, Hang Yan, Zili Wang, Tianyu Pang, and Michael Qizhe Shieh. Training optimal large diffusion language models. *arXiv:2510.03280*, 2025.

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, 2025.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, et al. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *arXiv:2504.16074*, 2025.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv:1806.03822*, 2018.

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&Q Benchmark. In *First Conference on Language Modeling*, 2024.

James V Roggeveen, Erik Y Wang, Will Flintoft, Peter Donets, Lucy S Nathwani, Nickholas Gutierrez, David Ettel, Anton Marius Graf, Siddharth Dandavate, Arjun Nageswaran, et al. Hardmath2: A benchmark for applied mathematics built by students as part of a graduate class. *arXiv:2505.11774*, 2025.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv:2210.01240*, 2022.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv:1909.08053*, 2019.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference, 2025.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv:2310.16049*, 2023.

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

Aider-AI team. Aider-ai/aider, 2023a. URL https://github.com/Aider-AI/aider.

Nexusflow.ai team. Nexusraven-v2: Surpassing gpt-4 for zero-shot function calling, 2023b. URL https://nexusflow.ai/blogs/ravenv2.

Opencompass team. Open-compass/opencompass, 2023c. URL https://github.com/open-compass/opencompass.

Changxin Tian, Jiapeng Wang, Qian Zhao, Kunlong Chen, Jia Liu, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. Wsm: decay-free learning rate schedule via checkpoint merging for llm pre-training. *arXiv:2507.17634*, 2025.

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

Peiding Wang, Li Zhang, Fang Liu, Lin Shi, Minxiao Li, Bo Shen, and An Fu. Codeif-bench: Evaluating instruction-following capabilities of large language models in interactive code generation. *arXiv:2503.22688*, 2025b.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv:2307.10635*, 2023.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion LLMs Can Do Faster-Than-AR Inference via Discrete Diffusion Forcing, August 2025c. arXiv:2508.09192.

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

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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. Cmath: Can your language model pass chinese elementary school math test? *arXiv:2306.16636*, 2023.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream-Coder 7B: An Open Diffusion Language Model for Code, September 2025. arXiv:2509.01142.

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

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

Runpeng Yu, Qi Li, and Xinchao Wang. Discrete Diffusion in Large Language and Multimodal Models: A Survey, September 2025. arXiv:2506.13759.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv:1809.08887*, 2018.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv:1905.07830*, 2019.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark. *arXiv:2305.12474*, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv:2311.07911*, 2023.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model, 2025.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv:2406.15877*, 2024.