LLaDA2.1: Speeding Up Text Diffusion via Token Editing

Tiwei Bie[1], Maosong Cao[1], Xiang Cao[1], Bingsen Chen[1], Fuyuan Chen[1], Kun Chen[1], Lun Du[1], Daozhuo Feng[1], Haibo Feng[1,4], Mingliang Gong[1], Zhuocheng Gong[1], Yanmei Gu[1], Jian Guan[1], Kaiyuan Guan[1], Hongliang He[1,3], Zenan Huang[1], Juyong Jiang[1], Zhonghui Jiang[1], Zhenzhong Lan[1,3,†], Chengxi Li[1], Jianguo Li[1,†], Zehuan Li[1], Huabin Liu[1], Lin Liu[1], Guoshan Lu[1], Yuan Lu[1], Yuxin Ma[1], Xingyu Mou[1], Zhenxuan Pan[1], Kaida Qiu[1], Yuji Ren[1], Jianfeng Tan[1], Yiding Tian[1], Zian Wang[1], Lanning Wei[1], Tao Wu[1], Yipeng Xing[1], Wentao Ye[1,2], Liangyu Zha[1], Tianze Zhang[1], Xiaolu Zhang[1], Junbo Zhao[1,2,†], Da Zheng[1,†], Hao Zhong[1,2], Wanli Zhong[1,4], Jun Zhou[1], Junlin Zhou[1], Liwang Zhu[1], Muzhi Zhu[1,2], Yihong Zhuang[1]

[1]Ant Group, [2]Zhejiang University, [3]Westlake University, [4]Southern University of Science and Technology

Abstract

While **LLaDA2.0** showcased the scaling potential of 100B-level block-diffusion models and their inherent parallelization, the delicate equilibrium between decoding speed and generation quality has remained an elusive frontier. Today, we unveil **LLaDA2.1**, a paradigm shift designed to transcend this trade-off. By seamlessly weaving **Token-to-Token (T2T)** editing into the conventional **Mask-to-Token (M2T)** scheme, we introduce a joint, configurable threshold-decoding scheme. This structural innovation gives rise to two distinct personas: the ***Speedy Mode (S Mode)***, which audaciously lowers the M2T threshold to bypass traditional constraints while relying on T2T to refine the output; and the ***Quality Mode (Q Mode)***, which leans into conservative thresholds to secure superior benchmark performances with manageable efficiency degrade. Furthering this evolution, underpinned by an expansive context window, we implement the first large-scale **Reinforcement Learning (RL)** framework specifically tailored for dLLMs, anchored by specialized techniques for stable gradient estimation. This alignment not only sharpens reasoning precision but also elevates instruction-following fidelity, bridging the chasm between diffusion dynamics and complex human intent. We culminate this work by releasing **LLaDA2.1-Mini (16B)** and **LLaDA2.1-Flash (100B)**. Across 33 rigorous benchmarks, LLaDA2.1 delivers strong task performance and lightning-fast decoding speed. Despite its 100B volume, on coding tasks it attains an astounding **892 TPS** on HumanEval+, **801 TPS** on BigCodeBench, and **663 TPS** on LiveCodeBench.

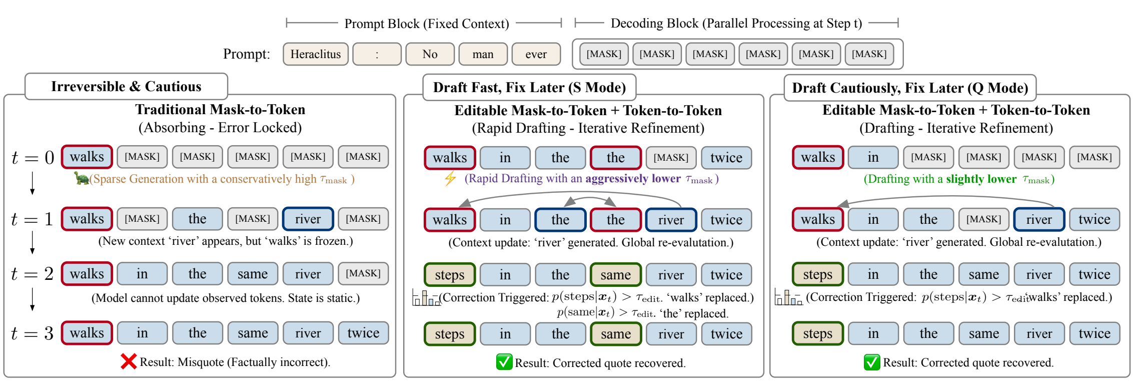

Figure 1: **Aggressive parallel drafting**, backed by retroactive correction, accelerates inference.

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

1 Introduction

Discrete diffusion Large Language Models (dLLMs) have emerged as a compelling alternative to autoregressive generation, offering the potential for non-monotonic reasoning and parallel decoding. However, the standard absorbing-state framework—which enforces a rigid, monotonic transition from [MASK] to fixed tokens—faces inherent limitations in fidelity. As highlighted by Kang et al. (2025), the independent nature of parallel decoding often amplifies token-level inconsistencies. While recent studies have attempted to mitigate this via confidence-based remasking (Wang et al., 2025b) or by employing external guide models (Lee et al., 2025). To bridge the gap between efficient parallel generation and high-fidelity reasoning, we align with the direction of generalizing discrete diffusion beyond absorbing states (Rütte et al., 2025) and propose a comprehensive framework for Editable State Evolution.

Unlike prior work such as Song et al. (2025), we first design a novel **Error-Correcting Editable** decoding strategy, which introduces a dynamic paradigm controlled by dual probability thresholds. This paradigm encompasses two types of operations: direct decoding from mask to token, and editing from one token to another. This strategy enables the model to directly refine its own outputs during the generation process, thereby effectively addressing the local inconsistencies commonly encountered in parallel decoding. To cultivate this editing capability, our CPT and SFT phases expose the model to both masked positions and stochastic noise, incentivizing it to not only generate new content but also identify and rectify existing errors.

Crucially, this architecture transforms the rigid trade-off between latency and fidelity into a flexible, user-configurable continuum. By allowing the model to retroactively correct errors, we can aggressively lower the confidence threshold for the initial Mask-to-Token (M2T) phase without collapsing the generation quality. This insight gives rise to two distinct operating personas: a ***Speedy Mode (S Mode)***, which prioritizes high-throughput generation by accepting lower-confidence tokens and relying on subsequent Token-to-Token (T2T) passes for rectification; and a ***Quality Mode (Q Mode)***, which adheres to conservative thresholds to maximize reasoning rigor. This duality demonstrates that editability is not merely a mechanism for error repair, but a fundamental lever for accelerating parallel decoding.

To further elevate the model's capabilities, we integrate a **Reinforcement Learning (RL)** stage. While recent works such as SPG (Wang et al., 2025a), TraceRL (Wang et al., 2025c) and ESPO (Ou et al., 2025) have demonstrated the potential of RL in improving dLLMs, applying policy gradients to block-autoregressive models remains challenging due to the intractability of sequence log-likelihoods. We circumvent this by adopting an ELBO-based Block-level Policy Optimization (EBPO) framework tailored for our editable setting.

Notice that LLaDA2.1 extends its previous version (LLaDA2.0) by prioritizing decoding versatility over mere parameter scaling or benchmark peaking. By keeping the model size constant and minimal change of training data, we prove that our novel editing scheme enables lightning-fast execution with minimal overhead. This work serves as a proof-of-concept for a new dLLM paradigm that balances high-quality generation with extreme operational efficiency.

2 Configurable Decoding Scheme

During LLM decoding, **Exposure Bias**—where errors compound as the model conditions on its own imperfect predictions—is inevitable. This phenomenon is particularly severe in dLLMs due to their parallel generation nature. We observe that once such decoding errors occur, dLLMs tend to become increasingly conservative in subsequent steps, significantly slowing down the generation process. In contrast, autoregressive models exhibit lower exposure bias and can self-correct through extended chain-of-thought reasoning. To address this challenge, we introduce an **editing** operation into the decoding process, enabling the model to retrospectively correct errors introduced during parallel generation, thereby achieving a much better balance between generation speed and quality.

Specifically, we extend standard discrete diffusion to support it. Unlike conventional absorbing-state models that enforce a rigid monotonic transition from [MASK] to fixed tokens, our framework introduces a dynamic "Draft-and-Edit" paradigm controlled by dual probability thresholds. We formalize the state evolution by defining two active update sets at timestep t: the *Unmasking Set* Γ_t and the *Editing Set* Δ_t.

We formalize the state evolution by defining two active update sets at timestep t: the *Unmasking Set* Γ_t and

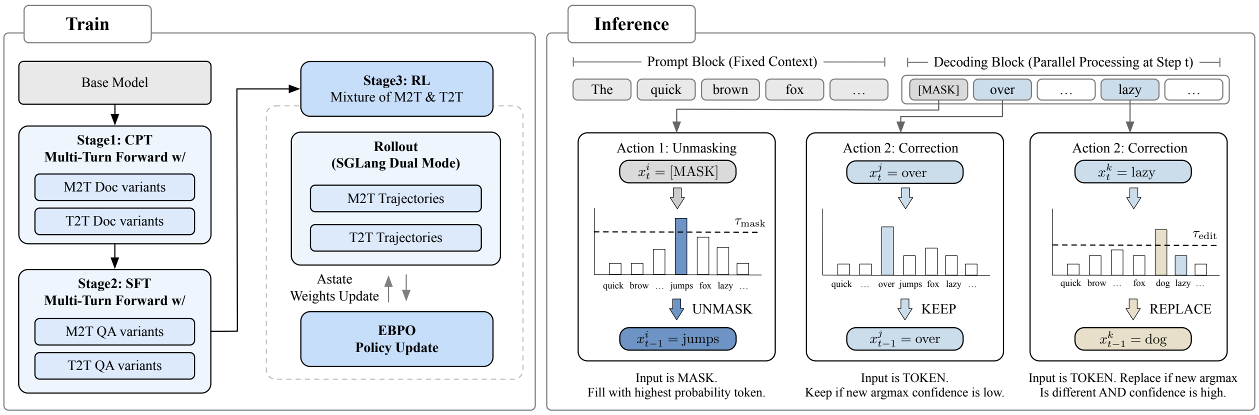

Figure 2: Overview of training & inference framework of LLaDA2.1

the *Editing Set* Δ_t. Let $v_t^i = \arg\max_v p_\theta(v|\boldsymbol{x}_t)$ be the top-candidate. The update indices are identified as:

$$\Gamma_t = \left\{ i \mid x_t^i = \text{[MASK] and } p_\theta(v_t^i|\boldsymbol{x}_t) > \tau_{\text{mask}} \right\}, \tag{1}$$

$$\Delta_t = \left\{ i \mid x_t^i \neq v_t^i \text{ and } p_\theta(v_t^i|\boldsymbol{x}_t) > \tau_{\text{edit}} \right\}, \tag{2}$$

with $\tau_{\text{mask}}, \tau_{\text{edit}} \in [0,1]$ being the confidence thresholds configuring the decoding dynamics. The transition operator then applies the updates strictly on the union of these sets:

$$x_{t-1}^i = \begin{cases} v_t^i & \text{if } i \in \Gamma_t \cup \Delta_t, \\ x_t^i & \text{otherwise.} \end{cases} \tag{3}$$

3 Training Paradigm

3.1 Training Alignment for "Draft-and-Edit"

To align the model with the "Draft-and-Edit" inference paradigm and mitigate the *Exposure Bias* inherent in standard mask-based training, we employ a unified **Mixture of M2T and T2T** objective. This objective is applied throughout both the Continual Pre-Training (CPT) and Supervised Finetuning (SFT) stages.

This dual-stream training objective enables the model to develop two complementary capabilities fundamental to our framework:

- **Drafting Stream (Mask-to-Token):** The model learns to predict the correct token at each masked position to generate initial content, establishing the foundational drafting capability.
- **Editing Stream (Token-to-Token):** The model learns to recover original tokens from random noise perturbations (rectifying errors), equipping it with the ability to identify and rewrite artifacts.

By consistently applying this dual-stream supervision from CPT through SFT, we ensure that LLaDA2.1 is fundamentally conditioned to function as both a fast drafter and a precise editor within a single parameter space. Additionally, we employ a Multi-turn Forward (MTF) data augmentation technique, by exposing the model to a wider variety of editing scenarios, enhance the model's editing capabilities.

3.2 Reinforcement Learning Training

The application of policy gradient methods to diffusion models faces a fundamental hurdle: the intractability of the sequence-level log-likelihood, $\log \pi_\theta(\boldsymbol{x})$, which is essential for computing policy updates. While prior works have explored various approximations, they have historically struggled with high variance and prohibitive computational costs, limiting RL to small-scale experiments (Wang et al., 2025c; Ou et al., 2025; Wang et al., 2025a). We overcome this bottleneck by synthesizing **ELBO-based Block-level Policy Optimization (EBPO)** with robust infrastructure optimizations. By utilizing the Evidence Lower Bound (ELBO) as a principled proxy for exact likelihood and implementing **Vectorized Likelihood Estimation** (Arriola et al., 2025) to parallelize bound computation, we achieve orders-of-magnitude acceleration. This integration allows us to scale dLLMs RL to unprecedented context lengths and training magnitudes, establishing a stable and efficient pipeline for post-training.

Formally, we maximize a clipped surrogate objective, where the advantage is weighted by the probability ratio ρ:

$$\mathcal{J}_{\text{EBPO}}(\theta) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\pi_{\theta_{\text{old}}}}\left[\min\left(\rho(\boldsymbol{y}|\boldsymbol{x})\hat{A}, \text{clip}(\rho(\boldsymbol{y}|\boldsymbol{x}), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\hat{A}\right)\right], \tag{4}$$

where $\hat{A}$ is an estimator of the advantage function at timestep t, quantifying the relative improvement of the chosen action over the average expectation under the current policy. For a set of discretized timesteps $\{t_n\}_{n=1}^{N}$ and weights $\{w_n\}$, we construct a composite input $\boldsymbol{z}_n = \boldsymbol{y}_{t_n} \oplus \boldsymbol{y}_0$ to compute all block-conditional probabilities in parallel:

$$\log \rho(\boldsymbol{y}|\boldsymbol{x}) \approx \sum_{n=1}^{N} w_n \sum_{b=1}^{B} \left(\log p_\theta(\boldsymbol{y}^b \mid \boldsymbol{z}_n, \boldsymbol{x}; \mathcal{M}) - \log p_{\theta_{\text{old}}}(\boldsymbol{y}^b \mid \boldsymbol{z}_n, \boldsymbol{x}; \mathcal{M})\right). \tag{5}$$

Here, $\mathcal{M}$ denotes a Block-Causal Mask ensuring the b-th block attends only to valid history. By aggregating block-level contributions ($\sum_{b=1}^{B}$) within a single forward pass per timestep n, we establish a computationally tractable pipeline for scaling reinforcement learning to long-context diffusion generation.

4 Infrastructure

4.1 Training Infrastructure

Continued Pre-Training and Supervised Fine-Tuning For both continued pre-training (CPT) and supervised fine-tuning (SFT), we adopt the same training infrastructure as LLaDA2.0 (Bie et al., 2025), leveraging dFactory (InclusionAI, 2025), which provides efficient training recipes specifically designed for dLLMs, except that we introduce a dedicated optimized implementation for the multi-turn forward (MTF) stage.

RL Training To enable effective policy optimization for dLLMs, we extend the AReaL framework (Fu et al., 2025; Mei et al., 2025) by developing specialized likelihood estimation and advantage estimation protocols that leverage diffusion sampling, explicitly supporting both T2T and M2T modes. This workflow is powered by ASystem (Ling Team et al., 2025) for distributed orchestration and utilizes a customized version of SGLang (Ant Group Team & SGLang Team) as the dedicated rollout engine.

4.2 Inference Infrastructure

We use a customized version of SGLang (Ant Group Team & SGLang Team) for inference. To further accelerate the inference speed, we integrate Alpha-MoE (Aleph-Alpha), a MoE megakernel that combines the two FusedMoE computations into one kernel, and adopt per-block FP8 quantization to balance the inference speed and model accuracy. To accelerate inference on long-context sequences, we adopt block-wise causal masked attention, allowing the KV cache for the entire long context to be computed in a single forward pass. We further enable radix caching and batching support for block diffusion LLMs in SGLang.

4.3 Decoding Algorithm at Inference

In the inference stage, we adopt a decoding algorithm that combines Threshold Decoding (Ma et al., 2025) with an explicit editing mechanism. In the basic setting, decoding and editing are performed within a single block: tokens are generated under a threshold-based constraint, and local edits are applied to revise intermediate outputs before the block is finalized.

Beyond single-block editing, we further introduce a **Multiple Block Editing (MBE)** mechanism. MBE allows the model to revisit and revise previously generated blocks based on the content of newly decoded blocks.

5 Evaluation

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

- **Knowledge**: MMLU-Pro (Wang et al., 2024), GPQA-Diamond (Rein et al., 2024), C-Eval (Huang 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 (Opencompass Team, 2023)

- **Coding**: CRUXEval (Gu et al., 2024), MultiPL-E (Cassano et al., 2023), BigCodeBench (Zhuo et al., 2024), LiveCodeBench (Jain et al., 2024), Spider (Yu et al., 2018), BIRD (Li et al., 2023), HumanEval+ (Liu et al., 2023), MBPP+ (Liu et al., 2023)
- **Math**: OlympiadBench (He et al., 2024), AIME 2025 (AIME, 2025), Omni-MATH (Gao et al., 2024), GSM-Plus (Li et al., 2024), CMATH (Wei et al., 2023)
- **Agent & Alignment**: BFCL (Patil et al., 2025), IFEval (Zhou et al., 2023), Nexus Function Calling Benchmark (Nexusflow.ai Team, 2023)

We report the comparative scores and TPF (tokens per forward) of LLaDA2.1-flash and LLaDA2.1-mini against other models in Tables 1 and 2, respectively. From the results, we observe that LLaDA2.1's scores under *S Mode* decrease compared to LLaDA2.0, but a substantial improvement in TPF is achieved. While under *Q Mode*, LLaDA2.1 surpasses the results of LLaDA2.0 on both mini and flash model.

In Table 3, we focus on showcasing the speed performance of LLaDA2.1 in *S Mode*. It can be observed that LLaDA2.1 exhibits significant speed variations across different domains, being highest in the code domain and lowest in instruction following. Specifically, after quantization, LLaDA2.1-flash achieves a peak TPS of 891.74 on HumanEval+, while LLaDA2.1-mini reaches 1586.93 in peak TPS, demonstrating significant speed advantages.

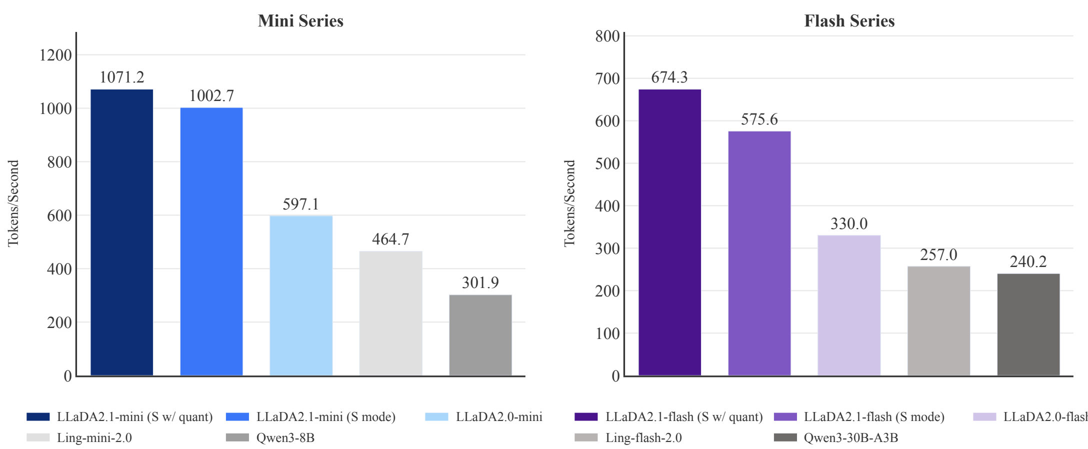

Figure 3: Throughput (TPS) comparison on nine benchmarks, consistent with the evaluation settings in Table 3, for LLaDA2.1 variants against LLaDA2.0, Ling, and Qwen3 across the mini (left) and flash (right) series.

As shown in Table 4, under the same *S Mode* setting, Multi-Block Editing (MBE) yields consistent performance improvements across benchmarks for both Flash and Mini variants, at the cost of a modest reduction in throughput. The gains are particularly evident on reasoning and coding tasks, indicating that iterative cross-block refinement effectively corrects local errors and improves global consistency without substantially compromising decoding efficiency.

Figure 3 further illustrates the throughput (in terms of token per sec) comparison of LLaDA 2.1 variants against LLaDA 2.0, Ling, and Qwen-3 across 5 different benchmark domains as shown in Table 3. This comparison spotlights LLaDA-2.1 (S Mode)'s striking speed advantage: it achieves dramatically faster inference while sacrificing only a negligible sliver of output quality.

6 Outlook and Limitation

Tradeoff Between Inference Speed and Accuracy While LLaDA2.1 significantly improves inference speed, a clear speed-accuracy tradeoff persists, particularly with noticeable performance differences across various

Table 1: Benchmark Performance of LLaDA2.1-flash, comparing with several baseline models. For diffusion language model, we report its scores across each benchmark along with its TPF (tokens per forward); for AR model, we report its scores only, as its TPF is inherently equal to 1.

Benchmark	Qwen3-30B-A3B-Inst-2507 (Score)	Ling-flash-2.0 (Score)	LLaDA2.0-flash (Score | TPF)	LLaDA2.1-flash (S Mode) (Score | TPF)	LLaDA2.1-flash (Q Mode) (Score | TPF)
Average	73.09	71.52	72.43 | 3.08	72.34 | 5.93	73.54 | 3.64
Knowledge					
GPQA	54.14	69.16	62.31 | 3.29	66.67 | 3.95	67.30 | 2.37
MMLU-Pro	74.21	77.55	74.79 | 2.36	75.31 | 4.43	76.59 | 2.62
C-EVAL	88.12	87.54	85.21 | 1.90	86.93 | 2.71	86.71 | 1.75
PHYBench	29.84	27.67	30.06 | 2.70	26.04 | 4.10	28.23 | 2.66
TriviaQA	65.61	69.76	66.88 | 1.94	72.55 | 4.30	72.93 | 2.92
Reasoning					
BIG-Bench Hard	85.54	89.36	86.75 | 2.66	87.82 | 5.61	88.69 | 3.28
BIG-Bench Extra Hard	37.80	23.24	27.86 | 4.60	33.51 | 5.04	35.77 | 3.17
bbh-zh	86.18	75.09	87.52 | 3.21	82.55 | 5.78	86.23 | 3.77
MuSR	79.15	82.72	80.48 | 1.70	80.10 | 2.90	79.84 | 1.85
ZebraLogic	90.97	87.60	82.30 | 2.74	84.20 | 5.80	88.90 | 3.26
PrOntoQA	97.12	97.88	96.50 | 2.64	95.00 | 9.23	97.00 | 5.73
PIQA	91.57	91.95	92.76 | 1.43	92.44 | 2.38	92.17 | 1.44
OCNLI	71.59	65.36	71.63 | 1.09	72.17 | 1.83	72.75 | 1.32
HellaSwag	86.31	81.59	84.97 | 1.26	85.60 | 2.31	85.31 | 1.51
KOR-Bench	69.20	69.44	63.04 | 3.44	62.80 | 4.97	65.12 | 2.77
DROP	87.57	88.32	87.90 | 2.26	87.55 | 5.40	87.86 | 2.53
SQuAD 2.0	89.51	81.32	90.00 | 3.10	90.65 | 5.01	90.80 | 3.90
Coding					
LiveCodeBench	46.42	52.48	42.51 | 4.23	44.05 | 6.48	45.37 | 3.80
CRUXEval-O	86.75	82.75	85.12 | 3.21	85.25 | 6.54	87.50 | 3.80
MBPP+	78.21	80.89	79.37 | 4.02	76.72 | 10.43	77.25 | 5.96
HumanEval+	87.88	87.58	88.41 | 6.45	89.63 | 13.81	89.63 | 9.18
MultiPL-E	70.67	65.76	74.87 | 3.14	70.89 | 7.77	73.34 | 4.33
BigCodeBench-Full	41.49	40.70	41.58 | 3.33	37.11 | 8.51	39.21 | 4.70
BIRD-SQL	47.75	47.49	45.76 | 2.16	42.18 | 5.09	44.04 | 2.95
Spider	81.79	80.58	82.49 | 4.42	79.18 | 8.74	81.04 | 5.70
Math					
AIME 2025	61.88	55.89	60.00 | 4.57	63.33 | 5.36	63.33 | 3.46
OlympiadBench	77.59	76.19	74.07 | 3.70	75.85 | 6.46	76.59 | 3.81
GSM-Plus	89.41	89.71	89.74 | 2.68	89.23 | 7.14	89.69 | 3.83
CMATH	96.58	96.52	96.90 | 2.17	96.54 | 4.84	96.63 | 2.65
Omni-MATH	54.00	53.00	50.30 | 3.39	52.30 | 6.01	54.10 | 3.50
Agent & Alignment					
IFEval-strict-prompt	83.73	81.15	82.62 | 1.47	83.36 | 2.24	83.55 | 1.41
BFCL v3	73.41	67.69	74.94 | 4.87	74.86 | 9.24	75.61 | 6.76
Nexus FC	49.93	36.25	50.45 | 5.53	44.83 | 11.29	47.65 | 7.38

Table 2: Benchmark Performance of LLaDA2.0-mini, comparing with several baseline models. For diffusion language model, we report its scores across each benchmark along with its TPF (tokens per forward); for AR model, we report its scores only, as its TPF is inherently equal to 1.

Benchmark	Qwen3-8B (no_think) (Score)	Ling-mini-2.0 (Score)	LLaDA2.0-mini (Score | TPF)	LLaDA2.1-mini (S Mode) (Score | TPF)	LLaDA2.1-mini (Q Mode) (Score | TPF)
Average	61.59	64.72	63.39 | 2.60	62.07 | 5.34	63.90 | 3.12
Knowledge					
GPQA	48.01	59.41	47.76 | 2.73	48.36 | 3.62	53.28 | 2.12
MMLU-Pro	65.83	67.18	64.27 | 2.15	63.42 | 4.22	64.84 | 2.41
C-EVAL	80.60	82.17	81.80 | 1.78	78.40 | 3.39	78.59 | 1.91
PHYBench	9.76	14.59	11.70 | 2.48	12.75 | 4.41	13.05 | 2.52
TriviaQA	52.51	55.63	51.33 | 1.54	53.33 | 3.21	54.24 | 2.02
Reasoning					
BIG-Bench Hard	79.48	83.70	78.21 | 2.36	78.42 | 5.02	80.58 | 2.86
BIG-Bench Extra Hard	18.27	14.81	16.47 | 2.03	15.30 | 3.19	15.78 | 1.66
bbh-zh	80.09	66.11	75.75 | 2.77	67.65 | 3.89	70.40 | 2.35
MuSR	70.02	71.36	71.48 | 1.45	70.43 | 2.48	71.89 | 1.56
ZebraLogic	37.48	79.85	64.20 | 2.30	68.50 | 5.38	77.10 | 2.93
PrOntoQA	93.12	96.06	86.00 | 2.36	87.50 | 4.86	84.50 | 2.73
PIQA	88.30	87.54	86.51 | 1.45	84.87 | 2.59	86.89 | 1.45
OCNLI	61.49	60.17	64.51 | 4.06	61.02 | 1.78	61.59 | 1.23
HellaSwag	79.56	69.02	79.01 | 1.50	75.71 | 2.39	76.19 | 1.49
KOR-Bench	54.96	63.20	49.92 | 2.45	46.64 | 4.28	48.00 | 2.35
DROP	84.56	78.80	81.91 | 2.02	81.55 | 5.84	82.37 | 2.87
SQuAD 2.0	85.21	75.56	86.50 | 2.47	84.51 | 4.33	85.13 | 3.09
Coding					
LiveCodeBench	26.76	42.29	31.83 | 3.34	28.85 | 6.42	30.40 | 3.63
CRUXEval-O	74.06	76.12	71.62 | 2.78	70.62 | 5.85	73.75 | 3.35
MBPP+	72.69	77.25	78.24 | 3.43	73.28 | 10.59	74.07 | 6.30
HumanEval+	79.50	80.03	81.40 | 5.16	80.49 | 12.32	82.93 | 7.77
MultiPL-E	61.70	67.09	67.46 | 2.78	64.16 | 7.23	67.17 | 4.01
BigCodeBench-Full	36.05	35.00	32.89 | 2.87	30.18 | 7.33	34.39 | 4.09
BIRD-SQL	36.11	39.67	39.34 | 1.96	37.32 | 4.48	38.40 | 2.42
Spider	72.80	76.43	76.76 | 3.93	75.78 | 7.98	77.55 | 5.48
Math					
AIME 2025	22.08	47.66	36.67 | 2.41	36.67 | 6.34	43.33 | 3.29
OlympiadBench	55.33	72.30	67.70 | 2.63	64.30 | 7.08	66.67 | 3.99
GSM-Plus	85.56	87.18	86.50 | 2.41	85.88 | 6.82	86.55 | 3.69
CMATH	95.42	96.40	95.72 | 1.98	95.63 | 4.94	94.99 | 2.56
Omni-MATH	33.20	48.80	41.70 | 2.57	41.70 | 6.41	43.60 | 3.56
Agent & Alignment					
IFEval-strict-prompt	84.29	76.16	80.78 | 1.24	81.33 | 1.83	83.18 | 1.25
BFCL v3	70.12	53.75	70.72 | 4.26	72.06 | 7.39	73.61 | 5.14
Nexus FC	37.71	34.38	35.18 | 4.06	31.59 | 8.27	33.69 | 4.91

Table 3: Throughput (TPS) and relative score changes of Flash and Mini variants across benchmarks. For each model family, the w/o Quant setting serves as the baseline. Cells under w/ Quant are vertically split into *TPS* | Δ*Score*.

Category	Benchmark	LLaDA2.1-flash		LLaDA2.1-mini	
		w/o Quant TPS	w/ Quant TPS | ΔScore	w/o Quant TPS	w/ Quant TPS | ΔScore
Coding	HumanEval+	746.66	891.74 | -3.04	1496.67	1586.93 | -0.61
	MBPP+	639.47	761.38 | -1.85	1286.96	1303.96 | +1.85
	CRUXEval-O	550.09	645.72 | -0.24	980.82	1063.94 | -1.00
	BigCodeBench-Full	691.14	801.48 | +1.06	1220.40	1307.45 | -0.09
	LiveCodeBench	571.60	663.39 | -1.76	1015.82	1102.92 | +1.98
Math	GSM-Plus	574.65	667.07 | -0.03	1080.51	1186.18 | -0.30
Knowledge	GPQA-Diamond	416.92	477.79 | -0.64	724.30	784.62 | -1.64
Instruction Following	IFEval	219.37	248.25 | +1.48	338.58	365.52 | -1.29
Reasoning	PrOntoQA	770.88	912.16 | -1.00	880.19	938.93 | -1.50

Table 4: Performance comparison of LLaDA2.1-flash and Mini variants with and without Multi-Block Editing (MBE) across benchmarks. Each cell reports *Score* | *TPF*.

Category	Benchmark	LLaDA2.1-flash		LLaDA2.1-mini	
		w/o MBE Score | TPF	w/ MBE Score | TPF	w/o MBE Score | TPF	w/ MBE Score | TPF
Knowledge	MMLU-Pro	75.31 | 4.43	75.90 | 3.88	63.42 | 4.22	63.10 | 3.66
	TriviaQA	72.55 | 4.30	72.45 | 4.28	53.33 | 3.21	53.41 | 3.14
Reasoning	bbh-zh	82.55 | 5.78	83.21 | 4.85	67.65 | 3.89	67.94 | 3.41
	ZebraLogic	84.20 | 5.80	88.20 | 5.03	68.50 | 5.38	70.00 | 4.62
Coding	LiveCodeBench	44.05 | 6.48	46.48 | 5.62	28.85 | 6.42	29.74 | 5.44
	CRUXEval-O	85.25 | 6.54	87.00 | 5.62	70.62 | 5.85	70.62 | 5.02
	BigCodeBench-Full	37.11 | 8.51	39.30 | 7.00	30.18 | 7.33	30.70 | 6.05
	Spider	79.18 | 8.74	80.58 | 8.33	75.78 | 7.98	76.67 | 7.59
Math	AIME 2025	63.33 | 5.36	70.00 | 4.71	36.67 | 6.34	36.67 | 5.25
Agent & Alignment	IFEval-strict-prompt	83.36 | 2.24	83.55 | 2.11	81.33 | 1.83	83.55 | 1.70
Average	–	70.69 | 5.82	72.67 | 5.14	57.63 | 5.25	58.24 | 4.59

domains. It is necessary to adjust threshold parameters for different domains to balance speed and accuracy. In structured-data fields such as code and math, setting *S Mode* achieves high speed with little accuracy loss. However, in some general chat cases, these settings can cause undesirable output. In such cases, we recommend adjusting the parameters to *Q Mode*. Our conjecture is that this pattern may be related to the model's inherent preference for structured data or the distributional characteristics of training dataset. Further validation will be conducted in our future research.

Editable Enhanced dLLM Although dLLMs inherently support high parallelism, theoretically offering speed advantages over AR models, our experimental observations show that this high parallelism also introduces a higher error rate compared to AR models. These hidden errors can reduce the model's confidence in subsequent reasoning, ultimately slowing down the overall process. Therefore, timely editing to correct errors is essential. In our case analysis of LLaDA2.1, we observed that prompt editing corrected decoding errors, helping to maintain higher inference speeds. However, research on the editing capabilities of dLLMs is still in its early stages. We anticipate that future work, such as integrating editing into reinforcement learning, will further enhance the performance of editable dLLMs.

LLaDA2.1 remains in an experimental phase. Although rare, certain edge cases may occur. Empirical observations show that aggressively lowering the masking threshold τ_{mask} can quickly generate "rough drafts". Although the model's self-correction can partially alleviate the "stuttering" artifacts (such as n-gram repetitions) caused by independent parallel sampling, balancing drafting speed with the quality of the initial structure remains a key operational frontier. Overall, by unifying dynamic inference, hybrid training, and principled reinforcement learning, our work establishes a solid foundation for self-correcting discrete diffusion language models.

Conclusion Overall, LLaDA2.1 introduces an editing feature, which, through cumulative error correction, significantly lowered the decoding threshold of the dLLM and yielded considerable inference speed benefits. However, this model still faces many unresolved issues, and we anticipate that more powerful editable dLLMs will deliver even more unexpected and impressive results.

References

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

Aleph-Alpha. Alpha-MoE: A megakernel for faster tensor parallel inference. URL https://aleph-alpha.com/alpha-moe-a-megakernel-for-faster-tensor-parallel-inference/.

Ant Group Team and SGLang Team. Power Up Diffusion LLMs: Day-0 Support for LLaDA 2.0 | LMSYS Org. URL https://lmsys.org/blog/2025-12-19-diffusion-llm.

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.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Ling Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling Up Diffusion Language Models to 100B, December 2025.

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.

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

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale asynchronous reinforcement learning system for language reasoning, 2025. URL https://arxiv.org/abs/2505.24298.

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 preprint arXiv:2410.07985*, 2024.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. Crux-Eval: A Benchmark for Code Reasoning, Understanding and Execution. *arXiv preprint 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 preprint arXiv:2402.14008*, 2024.

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

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 preprint 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 preprint arXiv:1705.03551*, 2017.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, and Kangwook Lee. ParallelBench: Understanding the Trade-offs of Parallel Decoding in Diffusion LLMs, October 2025. URL http://arxiv.org/abs/2510.04767. arXiv:2510.04767 [cs].

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.

Sanghyun Lee, Sunwoo Kim, Seungryong Kim, Jongho Park, and Dongmin Park. Effective Test-Time Scaling of Discrete Diffusion through Iterative Refinement, November 2025. URL http://arxiv.org/abs/2511.05562. arXiv:2511.05562 [cs].

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, 2023.

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 preprint arXiv:2402.19255*, 2024.

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 preprint arXiv:2502.01100*, 2025.

Ling Team et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model, 2025. URL https://arxiv.org/abs/2510.18855.

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.

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 preprint arXiv:2410.06526*, 2024.

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

Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huanchen Zhang, and Yi Wu. Real: Efficient rlhf training of large language models with parameter reallocation. In *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. mlsys.org, 2025.

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

Opencompass Team. open-compass/opencompass, 2023. URL https://github.com/open-compass/opencompass.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for Diffusion LLMs Emerges from a Sequence-Level Perspective, December 2025. URL http://arxiv.org/abs/2512.03759. arXiv:2512.03759 [cs].

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 preprint arXiv:2504.16074*, 2025.

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

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, et al. Generalized Interpolating Discrete Diffusion. June 2025. URL https://openreview.net/forum?id=rvZv7sDPV9.

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

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, et al. 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 preprint arXiv:2310.16049*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, 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.

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 preprint arXiv:2510.09541*, 2025a.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking Discrete Diffusion Models with Inference-Time Scaling. October 2025b. URL https://openreview.net/forum?id=IJryQAOy0p.

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*, 2025c.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, 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 preprint arXiv:2306.16636*, 2023.

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 preprint arXiv:1809.08887*, 2018.

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

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, et al. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023.

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 preprint arXiv:2406.15877*, 2024.