dLLM: Simple Diffusion Language Modeling

Zhanhui Zhou* UC Berkeley
Lingjie Chen* UIUC
Hanghang Tong UIUC
Dawn Song UC Berkeley

Abstract

Although diffusion language models (DLMs) are evolving quickly, many recent models converge on a set of shared components. These components, however, are distributed across ad-hoc research codebases or lack transparent implementations, making them difficult to reproduce or extend. As the field accelerates, there is a clear need for a unified framework that standardizes these common components while remaining flexible enough to support new methods and architectures.

To address this gap, we introduce **dLLM**, an open-source framework that unifies the core components of diffusion language modeling—training, inference, and evaluation—and makes them easy to customize for new designs. With **dLLM**, users can reproduce, finetune, deploy, and evaluate open-source large DLMs such as LLaDA and Dream through a standardized pipeline. The framework also provides minimal, reproducible recipes for building small DLMs from scratch with accessible compute—including converting any BERT-style encoder or autoregressive LM into a DLM. We also release the checkpoints of these small DLMs to make DLMs more accessible and accelerate future research.

dLLM: https://github.com/ZHZisZZ/dllm
dllm-collection: https://huggingface.co/dllm-collection

1 Introduction

Diffusion language models (DLMs) have emerged as a promising alternative to standard autoregressive language modeling (Austin et al., 2021a; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Arriola et al., 2025), enabling iterative refinement (Wang et al., 2025; Havasi et al., 2025), flexible steering (Li et al., 2022; Schiff et al., 2025) and efficient decoding (Wu et al., 2025b;a; Ma et al., 2025; Ben-Hamu et al., 2025). Alongside this rapid progress, a growing number of open-weight DLMs have appeared (Nie et al., 2024; 2025; Ye et al., 2025; Chandrasegaran et al., 2025; Bie et al., 2025), and many of them share similar design choices. However, these common components are frequently distributed across ad-hoc research codebases, or lack transparent implementations, making them difficult to reproduce, compare, or extend.

To address this critical gap, we introduce **dLLM**, an open-source framework that standardizes the end-to-end development pipeline for diffusion language modeling around three core components: **training**, **inference**, and **evaluation**. (1) For training, **dLLM** provides unified trainer modules that cover the most common objectives in DLMs, including Masked Diffusion (Sahoo et al., 2024) and Block Diffusion (Arriola et al., 2025), while keeping diffusion modeling logic decoupled from model architectures so that new objectives and variants can be added with minimal refactoring. In practice, this enables users to reproduce and finetune existing DLMs (e.g., LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025)) and develop new models from scratch. (2) For inference, **dLLM** introduces a lightweight abstraction that enables plug-and-play inference algorithms (including optimized efficient decoding algorithms (Wu et al., 2025b)) without modifying existing model implementations. (3) For evaluation, **dLLM** provides a unified evaluation interface for reproducing official results across models.

*Equal contribution. Correspondence to zhanhui@berkeley.edu.

Beyond unifying existing DLM development pipelines, **dLLM** provides minimal, reproducible recipes for building small DLMs with accessible compute. These recipes include transparent end-to-end pipelines for converting existing LMs (e.g., BERT-style encoders (Devlin et al., 2019) and autoregressive language models (Gong et al., 2025)) into DLMs. We release checkpoints for these small models to support future research.

The key contributions of this work are:

- We introduce **dLLM**, an open-source framework that unifies the core components of diffusion language modeling—training, inference, and evaluation—in a standardized, modular and extensible workflow, enabling transparent development and faster iteration across new designs.
- We release minimal, end-to-end recipes and checkpoints for training small DLMs from scratch (e.g., converting BERT-style encoders and autoregressive LMs into DLMs), providing accessible starting points and baselines for future research.

2 Preliminaries

We denote a sequence of discrete tokens as $x = (x^1, \ldots, x^L) \in \mathcal{V}^L$, where $\mathcal{V}$ is a finite vocabulary. We introduce a continuous time variable $t \in [0, 1]$ and a special mask token $m \notin \mathcal{V}$. The clean data is denoted x_0, and x_t represents the corrupted sequence at time t.

Discrete Diffusion. Discrete diffusion models (Austin et al., 2021a; Sahoo et al., 2024; Lou et al., 2024) generate data by reversing a forward process that progressively destroys information. The forward process $q(x_t|x_0)$ adds noise (e.g., random masking) over time $t : 0 \to 1$, transforming the data into an uninformative state x_1. The generative reverse process $p_\theta(x_s|x_t)$ (where $s < t$) learns to denoise x_t to recover x_0. Unlike continuous diffusion, the state space remains discrete.

Masked Diffusion (MDLM). Masked Diffusion (MDLM) (Sahoo et al., 2024; Shi et al., 2024) simplifies the forward process as an absorbing-state masking process. In the forward process, each token x_0^i is independently masked with probability t, assuming linear schedule:

$$q(x_t^i|x_0^i) = (1-t)\mathbb{I}(x_t^i = x_0^i) + t\mathbb{I}(x_t^i = m), \tag{1}$$

where $\mathbb{I}$ is the indicator function. The model $p_\theta(x_0|x_t)$ is trained to predict the unmasked tokens at indices $\mathcal{M}_t$ where $x_t^i = m$. The training objective minimizes the negative log-likelihood of the clean tokens given the masked input, with a time-dependent reweighting (e.g., $1/t$ assuming linear schedule) to balance contributions across noise levels:

$$\mathcal{L}_{\text{MDLM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1),x_0}\left[\frac{1}{t}\sum_{i\in\mathcal{M}_t} -\log p_\theta(x_0^i|x_t)\right]. \tag{2}$$

Block Diffusion (BD3LM). Block Diffusion (BD3LM) (Arriola et al., 2025) combines autoregression with diffusion. The sequence x is partitioned into K non-overlapping blocks $B_1, \ldots, B_K$. We write x^{B_k} to denote the tokens in block B_k, and $x_t^{B_k}$ for its corrupted version at time t. The model generates blocks autoregressively, but each block is generated via a diffusion process conditioned on the clean history of previous blocks $x_{<B_k}$. The joint probability factorizes as $p_\theta(x) = \prod_{k=1}^K p_\theta(x^{B_k} \mid x^{<B_k})$. For a specific block B_k, the objective is the diffusion loss averaged over time, strictly applied to the current block tokens while freezing the history. We define $\text{mask}(B_k, t)$ as the set of masked indices within block B_k at time t, i.e., $\text{mask}(B_k, t) := \mathcal{M}_t \cap B_k$:

$$\mathcal{L}_{\text{BD3LM}} = \sum_{k=1}^K \mathbb{E}_{t\sim\mathcal{U}(0,1),x_0}\left[\frac{1}{t}\sum_{i\in\text{mask}(B_k,t)} -\log p_\theta(x_0^i \mid x_t^{B_k}, x^{<B_k})\right]. \tag{3}$$

This factorization allows the model to leverage cached key-values from previous blocks while generating the current block in parallel.

3 dLLM Overview

In this section, we provide an overview of the three core components of **dLLM**: Trainer (Section 3.1), Sampler (Section 3.2) and Evaluation (Section 3.3).

3.1 Trainer

```
# MDLM PT
trainer = MDLMTrainer(
  model=model,
  tokenizer=tokenizer,
  train_dataset=dataset["train"],
  eval_dataset=dataset["test"],
  args=training_args,
  data_collator=(
    DataCollatorForSeq2Seq()
  )
)
trainer.train()
```

(a) MDLM (e.g., LLaDA, Dream) Pretraining.

```
# MDLM PT -> BD3LM PT
trainer = MDLMTrainer(
trainer = BD3LMTrainer(
  model=model,
  tokenizer=tokenizer,
  train_dataset=dataset["train"],
  eval_dataset=dataset["test"],
  args=training_args,
  data_collator=(
    DataCollatorForSeq2Seq()
  )
)
trainer.train()
```

(b) Changes for BD3LM Pretraining.

```
# MDLM PT -> SFT
trainer = MDLMTrainer(
  model=model,
  tokenizer=tokenizer,
  train_dataset=dataset["train"],
  eval_dataset=dataset["test"],
  args=training_args,
  data_collator=(
+   NoAttentionMaskWrapper(
      DataCollatorForSeq2Seq(
+       label_pad_token_id=eos_token_id,
      ),
+   )
  ),
)
trainer.train()
```

(c) Changes for MDLM SFT.

```
# MDLM PT -> AR-to-MDLM Adapation
trainer = MDLMTrainer(
  model=model,
  tokenizer=tokenizer,
  train_dataset=dataset["train"],
  eval_dataset=dataset["test"],
  args=training_args,
+ right_shift_logits=True,
  data_collator=(
+   PrependBOSWrapper(
      DataCollatorForSeq2Seq(),
+   )
  ),
)
trainer.train()
```

(d) Changes for AR-to-MDLM adaptation.

Figure 1: **A unified trainer interface supports different purposes via modular trainers and configuration changes.** Figure 1a shows the MDLM pretraining setup. Figure 1b shows the single-line trainer swap from `MDLMTrainer` to `BD3LMTrainer`. Figure 1c shows the minimal changes to use `MDLMTrainer` for SFT: `NoAttentionMaskWrapper` keeps padding EOS visible, and `label_pad_token_id=eos_token_id` trains the model to generate EOS from extra mask tokens in inputs. Figure 1d shows the minimal changes to adapt an autoregressive LM to MDLM: `right_shift_logits` reuses next-token prediction, and `PrependBOSWrapper` prepends EOS to provide the predictions for the first mask token.

Unified training interface with Trainer (Figure 1). Most open-weight DLMs to date are trained with Masked Diffusion (MDLM) (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025) or Block Diffusion (BD3LM) (Arriola et al., 2025). Accordingly, the current version of **dLLM** focuses on unified `MDLMTrainer` and `BD3LMTrainer` as core training modules (`dllm/core/trainers`) that support both pretraining and finetuning most DLMs. At the same time, the framework's modular design can be easily extended to new diffusion objectives. For example, **dLLM** also includes a reference implementation of an EditFlow (Havasi et al., 2025; Nguyen et al., 2025) trainer for text diffusion with parallel insertion, substitution, and deletion operations.

Modular design enables easy customization (Figure 1). Our training pipeline follows a modular design that allows core components to be reused and extended with minimal changes—improving both flexibility and readability. Figure 1 illustrates this modularity in practice: switching between MDLM/BD3LM pretraining, MDLM SFT, and AR-to-MDLM adaptation requires only localized changes (e.g., swapping the trainer, toggling a small set of arguments, or wrapping the data collator), without altering the overall pipeline.

Simple yet scalable training powered by 🤗 HF infrastructure. Our training pipeline builds directly on the HuggingFace ecosystem. We use `accelerate` to support diverse training configurations (e.g., FSDP (Zhao et al., 2023) and DeepSpeed (Rajbhandari et al., 2020) for distributed training), and `peft` for parameter-efficient finetuning. Our custom trainers (e.g., `MDLMTrainer`) are lightweight wrappers around the `transformers` Trainer (Wolf et al., 2020). By using these components as building blocks, the framework stays easy to learn—allowing users to focus on DLM-specific logic—while remaining scalable enough to support large-model pretraining and research experimentation.

3.2 Sampler

```
# Inference
sampler = MDLMSampler(model, tokenizer)
sampler = MDLMFastdLLMSampler(model, tokenizer)

terminal_visualizer = TerminalVisualizer(tokenizer)

messages = [{"role": "user", "content": "Write a python
↪  function."}]

inputs = tokenizer.apply_chat_template(messages,
↪  add_generation_prompt=True, tokenize=True)

outputs = sampler.sample(inputs, return_dict_in_generate=True)

terminal_visualizer.visualize(outputs.histories, rich=True)
```

Figure 2: **Inference pipeline: sampler swap from vanilla to FastdLLM MDLM sampler.**

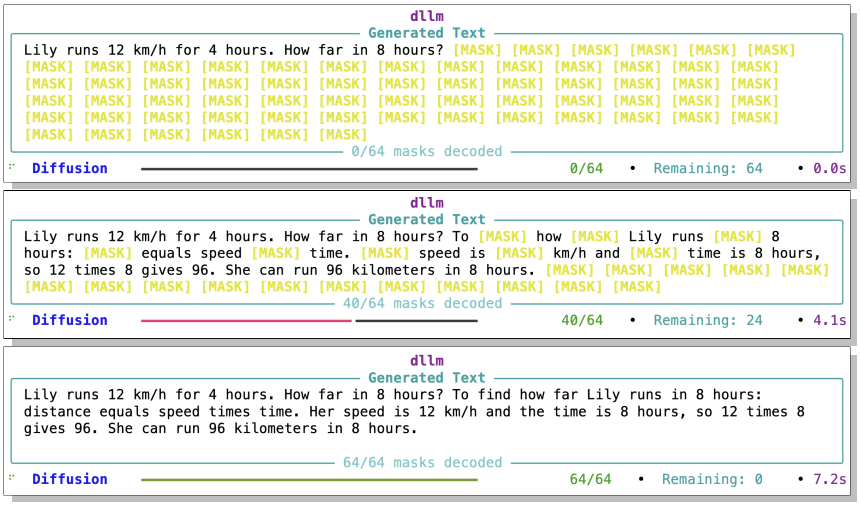


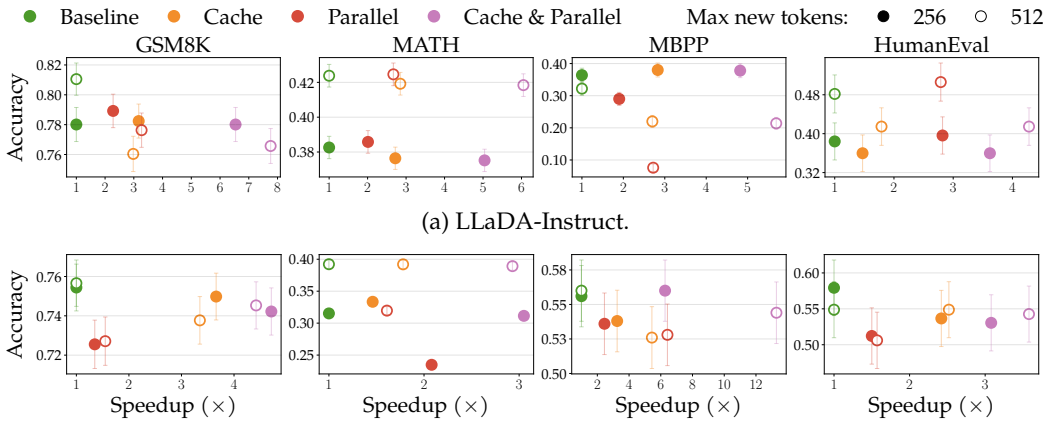
Figure 3: **Terminal Visualizer showing transition from masked to decoded tokens.**

Unified inference interface with Sampler (Figure 2). Different DLMs and inference algorithms expose inconsistent inference APIs, making it hard to reuse and compare inference

algorithms across models. To address this issue without modifying existing model implementations, we introduce a lightweight inference abstraction: `Sampler(model).sample()`. This wrapper decouples models from inference algorithms, allowing different samplers to be swapped in a plug-and-play manner while keeping the underlying model unchanged. Figure 2 illustrates the unified inference pipeline enabled by this interface.

Terminal visualizer (Figure 3). Unlike autoregressive LMs, which decode tokens strictly left-to-right, DLMs decode tokens in any order. As a result, the decoding order—beyond the final decoded output—is an important feature of dLLMs and is valuable for analysis. To support debugging and interpretability, we provide a terminal visualizer that reveals the token decoding order and the evolution of the sample over decoding steps (Figure 3).

Efficient DLM inference (Figure 2 & 4). DLM inference speed is a practical bottleneck (Wu et al., 2025b;a; Ma et al., 2025; Ben-Hamu et al., 2025). Building on the unified inference interface, the current version of **dLLM** includes an implementation of Fast-dLLM (Wu et al., 2025b) for accelerated MDLM decoding: `MDLMFastdLLMSampler`, which can be used as a drop-in replacement for the standard `MDLMSampler` (Figure 2). We report benchmarking results consistent with official Fast-dLLM implementations, demonstrating substantial inference speedups (see Figure 4 for visualization and Tables 6 and 7 in Appendix B for detailed results).



(a) LLaDA-Instruct.

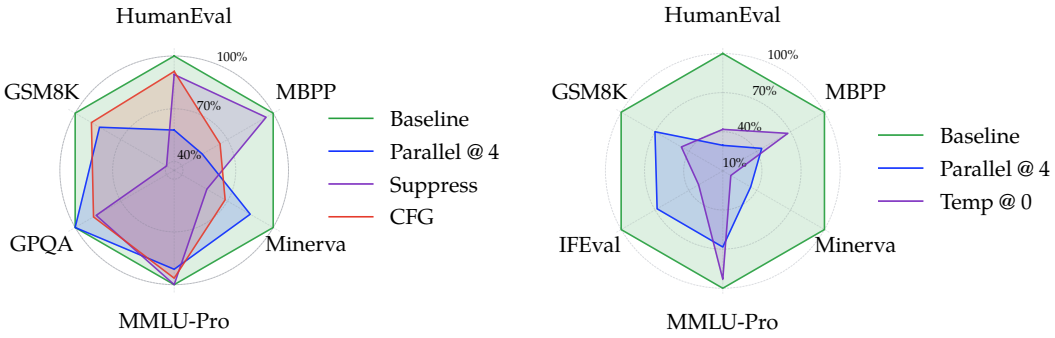
(b) Dream-Base.

Figure 4: **Fast-dLLM evaluation results with max new tokens @ 256 and 512.** Model selection follows the original Fast-dLLM evaluation protocol for consistency and fair comparison. textscCache uses block-wise approximate KV caching within each decoding block; PARALLEL uses confidence-based parallel token updates; CACHE & PARALLEL combines both. Note that max new tokens determines the number of pre-allocated padding tokens in the bidirectional context window, therefore affecting compute and measured performance.

3.3 Evaluation

Open-weight DLMs (Nie et al., 2025; Ye et al., 2025) rely on different evaluation tools, making unified evaluation difficult. This is further complicated by the fact that DLMs are especially sensitive to inference hyperparameters, as prior work often relies on task-specific hyperparameter tuning and postprocessing to achieve better scores. For example, even a single change in a key inference parameter can significantly alter performance (Figure 5).

A unified evaluation pipeline must therefore be flexible enough to support customization while faithfully reproducing the evaluation configurations used in prior work. To achieve this, we extend the `lm-evaluation-harness` (Gao et al., 2024) framework and carefully match the preprocessing, decoding settings, and post-processing used for each model–task pair with its corresponding official pipeline. These details vary across models and tasks and require manual verification, but this enables our framework to reproduce the reported,



(a) LLaDA-Instruct (b) Dream-Instruct

Figure 5: **Sensitivity to decoding hyperparameters.** We vary individual sampling hyperparameters at inference time and observe that performance can degrade sharply from the optimal configuration. BASELINE denotes the best-performing setting; SUPPRESS doesn't suppress <eos> from beginning of generation; CFG sets `cfg=0.5`; PARALLEL @ 4 generates four tokens per step; and TEMP @ 0 sets `temperature=0.0`.

model-specific scores while supporting consistent comparisons across models. Tables 4 and 5 (Appendix B) compare our reproduced results against the originally reported results, showing that our evaluation framework closely matches the official results.

Takeaway

DLM evaluation is highly sensitive to inference hyperparameters (e.g., max new tokens in Figure 4 and other settings in Figure 5)—a fact that is rarely explicitly stated in prior work, but is evident from our reproduction experiments.

4 Open DLMs with Open Recipes

Building on **dLLM**, we provide a set of fully reproducible recipes for training DLMs. These recipes cover (1) finetuning open-weight DLMs to reason (Section 4.1), and (2) training small DLMs from scratch with minimal compute (e.g., Section 4.2.1, Section 4.2.2). We make all of these model checkpoints available in 🤗 dllm-collection along with their evaluation results.

4.1 Finetuning Open-Weight Large DLMs

Training autoregressive LMs to reason before providing final answer has proven effective in solving complex tasks. Recent efforts such as d1 (Zhao et al., 2025) have begun exploring similar reasoning capabilities in DLMs. Using the unified trainer in **dLLM**, finetuning large DLMs is straightforward. We demonstrate that MDLM-style SFT can elicit reasoning capabilities in existing open-weight DLMs and improve their downstream performance.

Training details. We finetune both the Base and Instruct variants of LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) using MDLM SFT with LoRA on the s1K dataset (Muennighoff et al., 2025). Loss is computed only on response tokens. We use a maximum sequence length of 4096, 20 epochs, learning rate 10^{-5}, global batch size 8 with gradient accumulation steps of 4. We apply LoRA adaptation with $r = 128$, $\alpha = 256$, and weight decay 0.1. We adopt a cosine learning-rate schedule with 10% warmup. Training is conducted on 8×A100 GPUs using DeepSpeed ZeRO-2. See Figure 6 for training curves.

Evaluation results. We evaluate models SFTed on reasoning data by prepending a <reasoning> token at inference to force reasoning (Table 1). For Instruct models, reasoning SFT yields consistent gains across math, planning, and coding benchmarks. Base models show improvements on in-distribution math tasks (e.g., GSM8K (Cobbe et al., 2021),

Model	GSM8K	MATH500	Countdown	Sudoku	HumanEval	MBPP
LLaDA-Instruct	79.91	34.80	19.92	11.62	35.37	42.02
+ SFT (MDLM)	80.59	35.40	26.95	14.36	36.59	43.97
LLaDA-Base	64.67	10.20	10.16	0.34	25.00	40.08
+ SFT (MDLM)	73.62	17.40	8.98	0.00	18.90	28.79
Dream-Instruct	64.44	28.20	22.27	6.93	35.98	44.36
+ SFT (MDLM)	70.43	32.80	20.31	19.68	37.20	45.53
Dream-Base	49.05	20.40	10.55	1.07	15.85	22.96
+ SFT (MDLM)	63.00	23.40	8.59	1.03	29.88	23.35

Table 1: **MDLM SFT evaluation results**. Instruct models show consistent gains, Base models gain on in-distribution math but may regress on out-of-distribution planning and coding.

MATH500 (Hendrycks et al., 2021b)) but regress on out-of-distribution benchmarks. Overall, the results indicate that SFT is an effective starting point for reasoning in DLMs; all of these are achieved with the unified trainer interface (Figure 1) with little changes.

4.2 Training Small DLMs from Scratch

In addition to finetuning open-weight large DLMs, **`dLLM`** includes *recipes* and released *checkpoints* for training small DLMs from scratch (starting from backbones that are not DLMs). We cover two applications: (1) converting discriminative BERT models (Devlin et al., 2019) into DLMs and (2) converting autoregressive LMs into DLMs (Gong et al., 2025).

4.2.1 BERT-Chat: Converting BERTs to DLMs

Despite their traditional use in discriminative tasks, BERT-style models (Devlin et al., 2019) offer bidirectional representations well-suited for diffusive generation (Sahoo et al., 2024). We show that an off-the-shelf BERT-style model can be turned into a diffusion chatbot, without architectural changes, by finetuning only on instruction-following data. We build on top of the ModernBERT series (Warner et al., 2024) as the backbone, as they are among the strongest-performing BERT variants, and release two 🤗 checkpoints, ModernBERT-base-chat-v0.1 and ModernBERT-large-chat-v0.1.

Training details. We finetune ModernBERT-base and ModernBERT-large via MDLM SFT (no continual pretraining) on a mixture of instruction-tuning datasets: Tulu 3 SFT (Lambert et al., 2025) and SmolTalk (Ben Allal et al., 2025). The loss is computed only on response tokens. We use maximum sequence length 1024, 10 epochs, learning rate 10^{-4}, global batch size 384, bf16 precision, and a cosine learning-rate schedule with 10% warmup. Training runs on 8×A100 GPUs with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020). See Figure 7 for training curves.

Evaluation results. We evaluate BERT-Chats using **`dLLM`**'s unified evaluation pipeline (Table 2). A gap remains compared to decoder-only ARLMs of similar size (e.g., Qwen1.5-0.5B (Bai et al., 2023) on MMLU (Hendrycks et al., 2021a) and HellaSwag (Zellers et al., 2019)), yet the results are still noteworthy: ModernBERT-large-chat surpasses both GPT-2 (Radford et al., 2019) variants by a wide margin and outperforms Qwen1.5-0.5B-Chat on BBH (Suzgun et al., 2023) and MATH (Hendrycks et al., 2021b)—despite being an encoder-only model with no architectural modification for generation. This suggests that BERT-style backbones are a viable, if under-explored, starting point for DLMs.

4.2.2 Tiny-A2D: Converting ARLMs to DLMs

Autoregressive language models (ARLMs) dominate open-ended text generation, but DLMs offer complementary benefits such as parallel decoding and iterative refinement. Prior work has explored AR-to-diffusion conversion to bootstrap ARLM training artifacts into

Model	GSM8K	BBH	MATH	MMLU	Hellaswag	LAMBADA	WinoGrande
ModernBERT-base-chat-v0.1	3.6	21.1	3.1	26.2	34.5	49.3	48.8
ModernBERT-large-chat-v0.1	9.3	25.6	3.6	29.6	40.9	46.3	49.0
Qwen1.5-0.5B	22.0	18.3	3.1	39.2	48.2	48.6	55.0
Qwen1.5-0.5B-Chat	11.3	18.2	2.1	35.0	36.9	41.2	52.0
GPT-2	0.7	6.9	1.8	22.9	31.1	46.0	51.6
GPT-2-medium	2.1	17.8	1.4	22.9	39.4	55.5	53.1

Table 2: **ModernBERT-Chat evaluation results.** ModernBERT-Chat (Warner et al., 2024) and GPT-2 (Radford et al., 2019) models are evaluated with **dLLM**'s pipeline; Qwen1.5 (Bai et al., 2023) numbers are reported by original sources. See Figure 7 for training curves.

Model	GSM8K	BBH	MATH	MMLU	MMLU-Pro	Hellaswag	HumanEval	MBPP
Qwen3-0.6B-mdlm-v0.1	29.3	26.7	8.7	40.0	17.3	42.1	30.5	29.2
Qwen3-0.6B-bd3lm-v0.1	46.3	26.6	12.9	39.1	13.8	39.3	46.3	38.2
Qwen2.5-0.5B	41.6	20.3	19.5	47.5	15.7	52.1	30.5	39.3
Qwen3-0.6B-Base	59.6	41.5	32.4	52.8	24.7	47.4	32.3	36.6

Table 3: **Qwen-A2D evaluation results.** MDLM and BD3LM models are evaluated with **dLLM**'s pipeline; Autoregressive Qwen2.5/Qwen3 baselines are reported by original sources (Yang et al., 2025b;a). See Figure 8 for training curves.

DLMs (e.g., RND1 (Chandrasegaran et al., 2025) and DiffuLLaMA (Gong et al., 2025)). We show that an off-the-shelf ARLM can be converted into a diffusion chatbot with minimal changes: we take Qwen3-0.6B (Yang et al., 2025a) as the backbone and tune it under two diffusion objectives—MDLM (masked diffusion) (Sahoo et al., 2024) and BD3LM (block diffusion) (Arriola et al., 2025)—on instruction-following data. We release two 🤗 checkpoints, Qwen3-0.6B-diffusion-mdlm-v0.1 and Qwen3-0.6B-diffusion-bd3lm-v0.1.

Training details. We train both variants with only SFT (no continual pretraining) on the mixture of Tulu 3 SFT (Lambert et al., 2025), SmolTalk (Ben Allal et al., 2025), and opc-sft-stage1&2 (Huang et al., 2025). Loss is computed only on response tokens and we do not apply the logits right shifting tricks as in prior work (Gong et al., 2025; Chandrasegaran et al., 2025), because in our experiments this leads to performance degradation.. For the MDLM variant we use maximum sequence length 1024; for the BD3LM variant we use length 512 and block size 32. Both use 10 epochs, learning rate 10^{-4}, global batch size 2048, bf16 precision, and a cosine learning-rate schedule. Training is run on 64×A100 GPUs with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020). See Figure 8 for training curves.

Evaluation results. We evaluate both converted models using **dLLM**'s unified evaluation pipeline (Table 3). The BD3LM variant shows particular strength on code generation, with HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021b) scores that surpass the original Qwen3-0.6B-Base (Yang et al., 2025a) despite being trained with SFT alone. Overall, both DLM variants still trail their AR counterparts on most knowledge and reasoning benchmarks (e.g., MMLU (Hendrycks et al., 2021a), BBH (Suzgun et al., 2023)), reflecting the expected gap at this scale. Nonetheless, the fact that a competitive DLM can be obtained from an off-the-shelf ARLM with only SFT and no continual pretraining demonstrates that AR-to-diffusion conversion is a practical and compute-efficient path to building DLMs.

Takeaway

Existing pretrained models—both BERT-style encoders and autoregressive LMs—can be converted into functional DLMs with only minimal compute (e.g., supervised finetuning) and no architectural modification or continual pretraining, making DLM development accessible with minimal compute. Recent work (Fu et al., 2025) also validates that such lightweight conversion can match or surpass both the original AR models and DLMs trained from scratch.

5 Related Work

Discrete diffusion for text. Diffusion models, originally developed for continuous domains (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), were extended to discrete text via absorbing-state (D3PM (Austin et al., 2021a)) and uniform-state (multinomial diffusion (Hoogeboom et al., 2021)) formulations. Continuous-time extensions (Campbell et al., 2022), score-based (Sun et al., 2022; Meng et al., 2022), and ratio-based (Lou et al., 2024) objectives further unified the theory. Masked diffusion language models (MDLMs) simplify the forward process to independent token masking, with recent work clarifying equivalences and simplifying training (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024; Zheng et al., 2024). Alternative directions include continuous diffusion in embedding space (Li et al., 2022; Gong et al., 2023; Dieleman et al., 2022; Lin et al., 2023), flow matching and edit-based methods (Gat et al., 2024; Havasi et al., 2025; Nguyen et al., 2025), block diffusion (Arriola et al., 2025), which interpolates between AR and diffusion decoding for KV-cache reuse, and hybrid AR–diffusion architectures that use diffusion for speculative drafting (Christopher et al., 2025), planned outline-then-diffuse generation (Israel et al., 2025), or unified draft-and-verify passes (Liu et al., 2025).

Open-weight DLMs. Scaling DLMs has progressed rapidly. Nie et al. first scaled masked diffusion to 1.1B parameters. Converting pretrained autoregressive models into DLMs has proven effective: DiffuGPT/DiffuLLaMA adapt GPT-2 and LLaMA (127M–7B) (Gong et al., 2025); RND1 (Chandrasegaran et al., 2025) extends this to 30B; and LLaDA2.0 (Bie et al., 2025) scales to 100B with a 3-phase block-level scheme. At the 7–8B scale, Dream (Ye et al., 2025) adapts Qwen-2.5 with context-adaptive noise rescheduling, and LLaDA (Nie et al., 2025) trains an 8B MDLM from scratch, achieving performance competitive with LLaMA3-8B (Grattafiori et al., 2024). Commercial systems such as Mercury (Khanna et al., 2025) further demonstrate DLM viability in production.

Open tools for DLMs. Prior open-weight DLMs (Nie et al., 2025; Ye et al., 2025; Gong et al., 2025; Chandrasegaran et al., 2025; Bie et al., 2025) often lack unified development pipelines, making reproduction and comparison difficult. Open efficient inference tools for DLMs such as Fast-dLLM (Wu et al., 2025b) and Fast-dLLM v2 (Wu et al., 2025a) accelerate decoding but are developed independently of training and evaluation. Evaluation pipelines also vary across papers (e.g., task sets, inference hyperparameters, post-processing), and interfaces remain inconsistent. A framework unifying training, inference, and evaluation has been lacking. **dLLM** fills this gap with modular trainers, a plug-and-play sampler abstraction, and a reproducible evaluation pipeline aligned with official benchmarks (Section 3, Section 4).

6 Conclusion

We presented **dLLM**, an open-source framework that unifies the training, inference, and evaluation of DLMs in a modular, extensible pipeline. By standardizing the common components shared across recent DLMs, **dLLM** lowers the barrier to reproducing, finetuning, and fairly comparing existing models while making it straightforward to integrate new designs. Alongside the framework, we provide minimal recipes and checkpoints showing that existing pretrained models—both BERT-style encoders and autoregressive LMs—can be converted into competitive DLMs with lightweight finetuning alone, making DLM development increasingly accessible with minimal compute. We hope **dLLM** accelerates research and lowers the entry barrier for the broader community.

Future work. We plan to continue expanding **dLLM** by incorporating new methods as the field evolves, e.g., integrating RL algorithms once widely adopted approaches for DLMs emerge, and supporting additional open-weight models as they are released.

Acknowledgements

Zhanhui Zhou gratefully acknowledges support from the Berkeley Fellowship.

References

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. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021a.

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Lewis Tunstall, Colin Raffel, Leandro von Werra, Thomas Wolf, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 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, Lin 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. *arXiv preprint arXiv:2512.15745*, 2025.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 28266–28279, 2022.

Keshigeyan Chandrasegaran, Armin W. Thomas, Jerome Ku, Federico Berto, Jae Myung Kim, Garyk Brixi, Eric Nguyen, Stefano Massaroli, and Michael Poli. Rnd1: Simple, scalable ar-to-diffusion conversion. 2025. URL https://www.radicalnumerics.ai/blog/rnd1.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 12042–12059, 2025.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, et al. A framework for few-shot language model evaluation, 2024. URL https://zenodo.org/records/12608602.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations*, 2023.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *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 preprint arXiv:2407.21783*, 2024.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Flow matching with edit operations. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

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. In *NeurIPS Datasets and Benchmarks*, 2021b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12454–12465, 2021.

Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. OpenCoder: The open cookbook for top-tier code large language models. *arXiv preprint arXiv:2411.04905*, 2025.

Daniel Israel, Tian Jin, Ellie Cheng, Guy Van den Broeck, Aditya Grover, Suvinay Subramanian, and Michael Carbin. Planned diffusion. *arXiv preprint arXiv:2510.18087*, 2025.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2025.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022.

Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21051–21064. PMLR, 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. *arXiv preprint arXiv:2511.08923*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848. PMLR, 2024.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34532–34545, 2022.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

John Nguyen, Marton Havasi, Tariq Berrada, Luke Zettlemoyer, and Ricky T. Q. Chen. OneFlow: Concurrent mixed-modal and interleaved generation with edit flows. *arXiv preprint arXiv:2510.03506*, 2025.

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

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE/ACM, 2020.

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

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dallatorre, Bernardo P. de Almeida, Alexander M. Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2256–2265. JMLR.org, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 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.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Linguistics, 2020.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025a.

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

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 preprint arXiv:2505.09388*, 2025a.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025b.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch FSDP: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16 (12):3848–3860, 2023.

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

A Training Curves

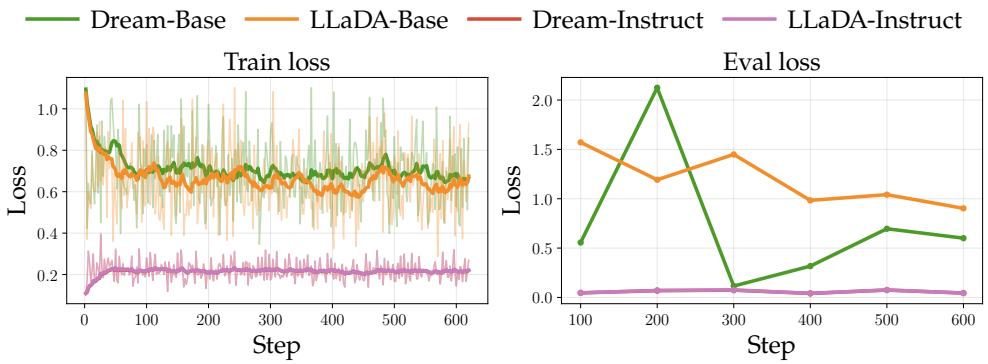


Figure 6: Training loss for finetuning open-weight DLMs to reason (Section 4.1).

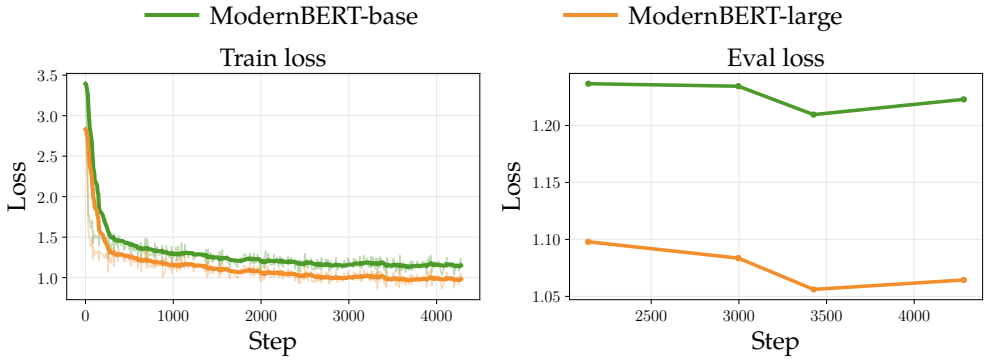


Figure 7: Training loss for finetuning BERT to chat (Section 4.2.1).

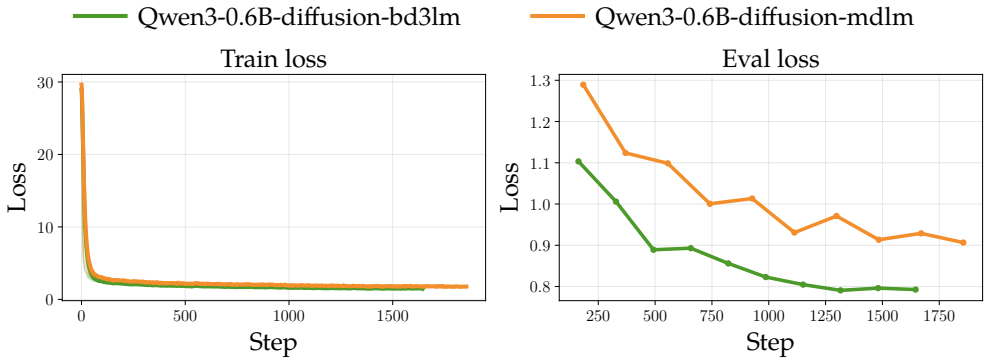


Figure 8: Training loss for finetuning autoregressive LMs to be DLMs (Section 4.2.2).

B Evaluation Reproduction

In this section, we report evaluation results comparing the official implementation (as reported in the original paper) with our unified **dLLM** reimplementation under the same configurations. Overall, our framework reproduces the official results closely across benchmarks, indicating that our evaluation pipeline and implementation are consistent with the official setup (with only minor necessary adjustments).

Tables 4 and 5 report our reproduced evaluation results for LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) with **dLLM**. Tables 6 and 7 report Fast-dLLM results, showing that our **dLLM** reimplementation achieves similar accuracy to the official numbers while substantially improving generation throughput.

Table 4: **LLaDA evaluation results.** "Official" denotes results from the original paper; "**dLLM**" denotes results from our **dLLM** reimplementation.

(a) LLaDA-Base

	MMLU	BBH	ARC-C	Hella.	WinoG.	PIQA	GSM8K	MATH	GPQA	Human.	MBPP
Official	65.9	49.7	45.9	70.5	74.8	73.6	70.3	31.4	25.2	35.4	40.0
dLLM	65.9	47.2	44.1	69.2	70.4	70.7	70.7	32.4	31.9	32.9	38.8

(b) LLaDA-Instruct

	MMLU	MMLU-Pro	ARC-C	Hella.	GSM8K	Math	GPQA	HumanE.	MBPP
Official	65.5	37.0	88.5	74.6	69.4	31.9	33.3	49.4	41.0
dLLM	69.8	36.2	86.4	76.7	74.7	31.9	30.6	47.0	40.0

Table 5: **Dream evaluation results.** "Official" denotes results from the original paper; "**dLLM**" denotes results from our **dLLM** reimplementation.

(a) Dream-Base

	MMLU	BBH	ARC-C	Hella.	WinoG.	PIQA	GSM8K	Math	GPQA	Human.	MBPP
Official	69.5	57.9	59.9	73.3	74.8	75.8	77.2	39.6	36.6	57.9	56.2
dLLM	70.0	63.7	59.0	73.5	72.5	76.4	77.0	42.4	34.6	56.7	56.0

(b) Dream-Instruct

	MMLU	MMLU-Pro	ARC-C	Hella.	GSM8K	Math	GPQA	HumanE.	MBPP
Official	67.0	43.3	—	—	81.0	39.2	33.0	55.5	58.8
dLLM	69.8	45.5	61.4	71.8	82.0	48.6	31.5	57.9	58.2

Table 6: **Fast-dLLM LLaDA-Instruct evaluation results with max new tokens @ 256 (a) and 512 (b).** "Official" denotes results from the Fast-dLLM paper; "**dLLM**" denotes results from our **dLLM** reimplementation.

(a) max new tokens @ 256

Benchmark	Source	Baseline		+Cache		+Parallel		+Both	
		Acc	Tok/s (×)	Acc	(×)	Acc	(×)	Acc	(×)
GSM8K	Official	79.3	6.7 (1.0×)	79.5	3.2×	79.2	2.5×	78.5	8.1×
	dLLM	78.0	8.1 (1.0×)	78.2	3.2×	78.9	2.3×	78.0	6.5×
MATH	Official	33.5	9.1 (1.0×)	33.3	2.6×	33.4	2.7×	33.2	5.7×
	dLLM	38.3	9.7 (1.0×)	37.6	2.7×	38.6	2.0×	37.5	5.0×
HumanEval	Official	41.5	30.5 (1.0×)	42.7	1.3×	43.9	3.3×	43.3	3.7×
	dLLM	38.4	18.8 (1.0×)	36.0	1.5×	39.6	2.8×	36.0	3.6×
MBPP	Official	29.4	6.0 (1.0×)	29.6	2.8×	28.4	4.1×	28.2	7.5×
	dLLM	36.4	9.3 (1.0×)	38.0	2.8×	29.0	1.9×	37.8	4.8×

(b) max new tokens @ 512

Benchmark	Source	Baseline		+Cache		+Parallel		+Both	
		Acc	Tok/s (×)	Acc	(×)	Acc	(×)	Acc	(×)
GSM8K	Official	77.5	3.2 (1.0×)	77.0	3.3×	77.6	5.8×	77.2	11.0×
	dLLM	81.1	6.7 (1.0×)	76.0	3.0×	77.6	3.3×	76.6	7.8×
MATH	Official	37.2	8.0 (1.0×)	36.2	2.5×	36.8	3.0×	36.0	5.9×
	dLLM	42.4	7.4 (1.0×)	41.9	2.9×	42.5	2.7×	41.8	6.0×
HumanEval	Official	43.9	18.4 (1.0×)	45.7	1.6×	43.3	3.1×	44.5	4.0×
	dLLM	48.2	13.0 (1.0×)	41.5	1.8×	50.6	2.8×	41.5	4.3×
MBPP	Official	14.8	4.3 (1.0×)	13.4	2.3×	15.0	5.1×	13.8	9.2×
	dLLM	32.2	7.7 (1.0×)	22.0	2.7×	7.6	2.7×	21.4	5.7×

Table 7: **Fast-dLLM Dream-Base evaluation results with max new tokens @ 256 (a) and 512 (b).** "Official" denotes results from the Fast-dLLM paper; "**dLLM**" denotes results from our **dLLM** reimplementation.

(a) max new tokens @ 256

Benchmark	Source	Baseline		+Cache		+Parallel		+Both	
		Acc	Tok/s (×)	Acc	(×)	Acc	(×)	Acc	(×)
GSM8K	Official	75.0	9.1 (1.0×)	74.3	3.6×	74.2	1.6×	74.8	5.3×
	dLLM	75.4	9.5 (1.0×)	74.3	3.6×	71.4	1.6×	76.2	5.1×
MATH	Official	38.4	11.4 (1.0×)	36.8	3.0×	37.9	2.4×	37.6	5.9×
	dLLM	29.3	25.1 (1.0×)	29.5	1.5×	24.3	2.1×	28.2	3.1×
HumanEval	Official	49.4	23.3 (1.0×)	53.7	1.5×	49.4	2.0×	54.3	2.8×
	dLLM	60.4	18.3 (1.0×)	57.3	1.9×	51.2	1.4×	57.3	2.7×
MBPP	Official	56.6	11.2 (1.0×)	53.2	3.1×	53.8	2.8×	56.4	6.8×
	dLLM	56.6	12.2 (1.0×)	51.6	2.7×	53.8	3.0×	55.6	6.0×

(b) max new tokens @ 512

Benchmark	Source	Baseline		+Cache		+Parallel		+Both	
		Acc	Tok/s (×)	Acc	(×)	Acc	(×)	Acc	(×)
GSM8K	Official	76.0	7.7 (1.0×)	74.3	3.3×	73.4	1.9×	74.0	5.6×
	dLLM	75.0	7.9 (1.0×)	75.1	3.3×	72.9	2.0×	76.4	5.4×
MATH	Official	39.8	9.6 (1.0×)	38.0	2.8×	39.5	3.2×	39.3	6.5×
	dLLM	37.5	17.7 (1.0×)	36.9	1.6×	31.5	2.2×	36.1	3.6×
HumanEval	Official	54.3	16.3 (1.0×)	54.9	1.7×	51.8	1.8×	54.3	3.2×
	dLLM	57.3	13.5 (1.0×)	56.1	2.0×	49.4	1.7×	54.9	3.5×
MBPP	Official	55.6	9.4 (1.0×)	53.8	2.8×	55.4	4.0×	55.2	7.8×
	dLLM	56.2	9.4 (1.0×)	52.8	2.7×	55.2	4.4×	55.2	7.7×