Data-Efficient Autoregressive-to-Diffusion Language Models via On-Policy Distillation

Xingyu Su[1*], **Jacob Helwig**[1*], **Shubham Parashar**[1*], **Atharv Chagi**[1], **Lakshmi Jotsna**[1], **Degui Zhi**[2], **James Caverlee**[1], **Dileep Kalathil**[1,3], **Shuiwang Ji**[1]

[1] Department of Computer Science and Engineering, Texas A&M University

[2] Department of Bioinformatics and Systems Medicine, University of Texas Health Science Center at Houston

[3] Department of Electrical and Computer Engineering, Texas A&M University

Abstract:

We study the transformation of autoregressive models (ARLMs) into diffusion language models (DLMs). Rather than pretraining from scratch, prior work replaces the causal attention in ARLMs with bidirectional attention and then trains the resulting model using a DLM objective. However, these approaches incur two distribution shifts. First, transitioning from a next-token prediction objective to a DLM objective can discard knowledge acquired by the ARLM during training. Second, standard DLMs suffer from a train-inference mismatch, as the training loss is defined on randomly masked sequences rather than the trajectories encountered at inference produced by confidence-based decoding. To address both challenges, we introduce an On-Policy Diffusion Language Model (OPDLM) in which On-Policy Distillation (OPD) is employed for ARLM-to-DLM transformation. Specifically, OPDLM is trained via self-OPD, where the student, an ARLM with bidirectional attention, generates its own trajectories, and the teacher, the original frozen ARLM, distills its knowledge by providing target logits on these trajectories. By training directly in an on-policy manner, OPDLM eliminates the train-inference mismatch in DLMs, while distillation from the original model enhances knowledge retention from the ARLM. Empirical results demonstrate that OPDLM requires **15**$\times$ to **7,000**$\times$ fewer training tokens with strong performance across a wide variety of tasks. OPDLM avoids the prohibitive cost of DLM pretraining and positions DLM transformation as a form of ARLM post-training.

Keywords: Diffusion Language Models, On-Policy Distillation, Knowledge Distillation, Language Models

***: These authors contributed equally**

Homepage: https://agenticscience.github.io/
Github Repository: https://github.com/divelab/OPDLM

Contents

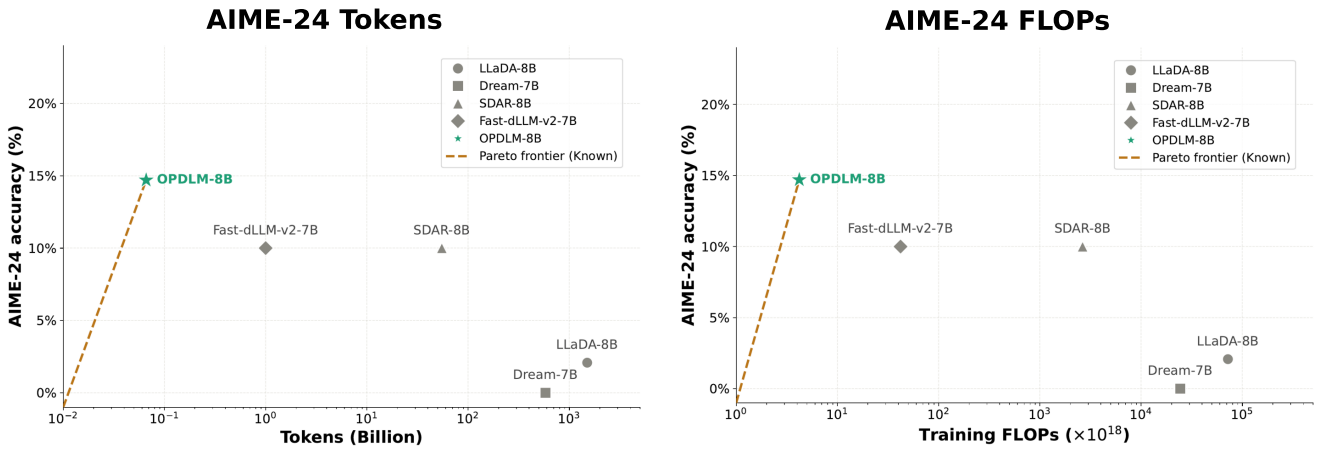

Figure 1: OPDLM establishes a new pareto frontier for AIME-24 [57]. Notably, training OPDLM-8B requires only 0.066B training tokens and 4.2×10^{18} FLOPs, which is $15\times$ to $7{,}000\times$ less as compared to established DLMs obtained from ARLMs [53, 8, 49], demonstrating its extreme compute and data efficiency.

1. Introduction

Diffusion Language Models (DLMs) extend diffusion-based generative modeling from domains such as images, video, robotics, and biology [22, 44, 12, 23, 35, 18] to text [3]. A prominent class of DLMs, Masked Diffusion Language Models (MDLMs) [41], has recently shown competitive performance with autoregressive language models (ARLMs) on arithmetic, coding, and general reasoning tasks [53, 32], while offering the potential advantage of decoding multiple tokens per step. Unlike ARLMs, which are trained by next-token prediction, MDLMs are pretrained by masking tokens in training sequences and minimizing a negative evidence lower bound (NELBO) [41]. Since pretraining MDLMs from scratch can require trillions of tokens [32, 2], recent work instead converts pretrained ARLM checkpoints into DLMs by replacing causal attention with bidirectional attention and continuing training with a diffusion objective, reducing the training requirement to billions of tokens [53, 8, 49]. However, this conversion process still leaves two mismatches that limit training efficiency.

The first is a knowledge-retention mismatch. A pretrained ARLM contains substantial knowledge acquired through next-token prediction, but replacing its attention structure and optimizing an MDLM objective can weaken or discard some of that knowledge. The second is a training–inference state mismatch: MDLMs are trained on uniformly random masked states, whereas inference follows model- and sampler-induced reverse unmasking trajectories, typically using confidence-guided decoding heuristics [32]. These trajectories are not directly represented by the standard forward-masking training distribution. This motivates two questions: *How can we retain knowledge from the original ARLM during conversion into a DLM?* and *how can we train the converted model directly on the states it visits during diffusion inference?*

In autoregressive settings, On-Policy Distillation (OPD) [1] offers a useful template for these goals. OPD supervises a student on states sampled from its own generation process, reducing exposure bias while distilling token-level knowledge from a teacher model. However, applying OPD to ARLM-to-DLM conversion is not straightforward, as standard OPD assumes the teacher and student share the same state space. For a DLM student, these states are partially masked diffusion states; therefore, a direct application would require a trained DLM teacher to provide targets on such states. This is precisely the requirement we aim to avoid.

In this work, we introduce On-Policy Diffusion Language Models (OPDLM), an on-policy method for

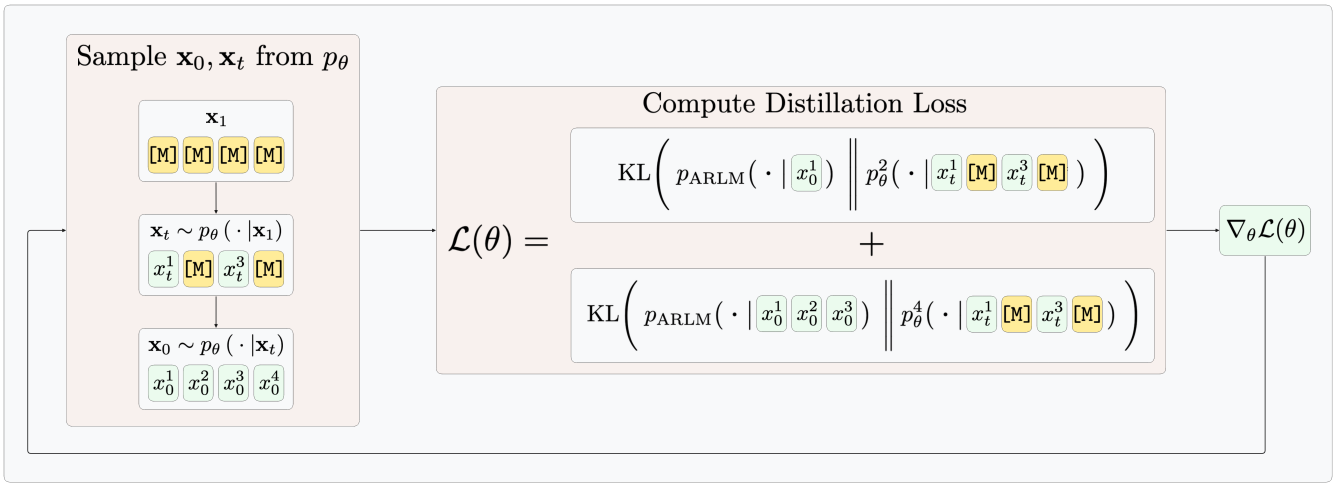

Figure 2: OPDLM training framework. At each training step, we sample a reverse decoding trajectory from the student DLM p_θ. From this trajectory, we select one partially denoised state $\mathbf{x}_t$ and compute a distillation loss over its masked tokens. For each masked position i, the loss aligns the student distribution p_θ^i with the frozen ARLM teacher distribution conditioned on the corresponding causal prefix of the terminal sequence $\mathbf{x}_0$. This trains the DLM on its own inference-time states while transferring knowledge from the original ARLM. For simplicity, the figure shows a single block of size 4, while in practice, trajectories contain multiple blocks.

converting ARLMs into DLMs. The DLM student is initialized from the ARLM checkpoint, samples its own reverse diffusion trajectories, and is trained on those trajectories using token-level targets from the original frozen ARLM. This combines on-policy training of the DLM student with knowledge distillation from the ARLM teacher, avoiding the need for a separately pretrained DLM teacher.

Our contributions are as follows. First, we formulate ARLM-to-DLM conversion as a post-training problem and show that DLMs can be obtained without the computationally intensive DLM pretraining stage. Second, we demonstrate strong data efficiency: OPDLM uses 15× to 7,000× fewer training tokens than established DLM baselines while remaining competitive across model scales and task categories (see Fig. 1). Third, we provide evidence that ARLM supervision supports knowledge retention during conversion, including zero-shot behavior on capabilities not explicitly targeted during conversion, such as multilingual generation and extended thinking. Finally, we show that the same framework can produce both general-purpose and task-specialized DLMs. We release our data, code, and model checkpoints to support future work.

2. Background and Related Works

Masked Diffusion Language Models (MDLMs) have emerged as an alternative to autoregressive models (ARLMs) for text generation [41, 32]. Structurally, MDLMs fall into two distinct paradigms: global diffusion across all masked positions [32, 41] and localized diffusion within a block-wise causal framework [2]. DLMs enable more flexible decoding and demonstrate superior training efficiency with improved convergence on perplexity [31, 16]. Despite these advantages, pretraining these models from scratch requires trillion-scale tokens [32].

ARLM to DLM Conversion is an efficient paradigm for pretraining DLMs [53, 49, 8, 45, 15, 60, 6]. Such pretraining reduces the required training data from trillions of tokens to billions, but the current conversion process is largely treated as a phase of continued pretraining [50, 30] in which the next-token prediction loss of ARLMs is replaced by the standard DLM loss with bidirectional attention. The adaptation of an ARLM

into a DLM has not been explored using knowledge distillation [21].

On-Policy Distillation (OPD) methods train a student model on trajectories sampled directly from its own policy, while a teacher model provides token-level supervision via KL divergence [1, 37, 10]. Unlike standard knowledge distillation [21], querying the teacher on these student-generated states has been shown to mitigate the distribution shift arising from learning with teacher generations [40]. Recently, self-OPD for ARLMs [58, 25, 42] has demonstrated that a single model can serve as its own teacher by leveraging privileged information to supervise a student conditioned on a strictly weaker context. In this work, we investigate whether OPD can be extended to train a student DLM with supervision from an ARLM teacher.

3. Preliminaries

In this section, we review masked diffusion language models (MDLMs) and On-Policy Distillation (OPD) for autoregressive language models (ARLMs).

Masked Diffusion Language Models (MDLMs). MDLMs define a forward masking process that corrupts a clean sequence[1] $\mathbf{x}_0 \sim p_{\text{data}}$, with $|\mathbf{x}_0| \le L$. For diffusion time $t \in [0,1]$, the forward process independently replaces each token with the MASK token $\mathbf{m}$:

$$q^{t|0}(x_t^i \mid x_0^i) = \alpha_t \delta_{x_0^i}(x_t^i) + (1-\alpha_t)\delta_{\mathbf{m}}(x_t^i), \qquad \alpha_t = 1 - t.$$

We use $q^{t|0}(\cdot \mid \mathbf{x}_0)$ to denote the corresponding product distribution over sequence-level corruptions. Thus, sampling $t \sim \mathcal{U}(0,1)$ corresponds to masking each token in $\mathbf{x}_0$ independently with probability t [32, 41]. For later comparison with on-policy training, it is useful to view this forward process as a distribution over trajectories. Given $\mathbf{x}_0$, let $\tau^{\text{fwd}} \sim \Gamma^{\text{fwd}}(\cdot \mid \mathbf{x}_0)$ denote a forward masking trajectory, viewed as a map from diffusion time to corrupted sequences, whose time-t marginal satisfies $\tau^{\text{fwd}}(t) \sim q^{t|0}(\cdot \mid \mathbf{x}_0)$. Standard MDLM training can then be viewed as sampling a trajectory τ^{fwd}, selecting a time $t \sim \mathcal{U}(0,1)$, and setting $\mathbf{x}_t = \tau^{\text{fwd}}(t)$. Given this corrupted sequence, the model is trained to recover the original tokens at masked positions as

$$\mathcal{L}_{\text{MDLM}}(\theta) = -\mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}},\, t \sim \mathcal{U}(0,1),\, \mathbf{x}_t \sim q^{t|0}(\cdot|\mathbf{x}_0)} \left[\frac{1}{t} \sum_{i \in \mathcal{M}(\mathbf{x}_t)} \log p_\theta(x_0^i \mid \mathbf{x}_t) \right], \tag{1}$$

where $\mathcal{M}(\mathbf{x}) = \{i : x^i = \mathbf{m}\}$ is the set of masked positions and the factor $1/t$ is the diffusion time weight induced by the linear schedule. Following [2], we use **block diffusion**, which partitions the sequence into B non-overlapping contiguous blocks. For block b, the model conditions on the clean prefix $\mathbf{x}_0^{<b}$ and the corrupted active block $\mathbf{x}_t^b$ as

$$\mathcal{L}_{\text{block}}(\theta) = -\mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}},\, t \sim \mathcal{U}(0,1),\, \mathbf{x}_t \sim q^{t|0}(\cdot|\mathbf{x}_0)} \left[\sum_{b=1}^{B} \frac{1}{t} \sum_{i \in \mathcal{M}(\mathbf{x}_t^b)} \log p_\theta(x_0^{b,i} \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b) \right]. \tag{2}$$

During inference, a new sequence is generated through a reverse unmasking procedure induced jointly by the model and sampler. Starting from an initially masked sequence, the sampler iteratively queries the model on the current partially denoised state, $p_\theta(\cdot \mid \mathbf{x}_{t_k})$, and selects and updates a subset of masked positions according to a decoding rule, typically based on model confidence. This process produces a reverse decoding trajectory $\tau := (\mathbf{x}_{t_T}, \mathbf{x}_{t_{T-1}}, \ldots, \mathbf{x}_{t_0})$, where $\mathbf{x}_{t_T}$ is the initial masked sequence, $\mathbf{x}_{t_0}$ is the terminal generated

[1] Unless otherwise stated, we write $\mathbf{x}$ for the sequence being denoised. For conditional generation, any prompt is treated as fixed observed context and omitted from the notation.

sequence, and the number of steps T and realized effective denoising times $t_T, t_{T-1}, \ldots, t_0$ are induced by the realized interaction between the model and sampler. We view τ as a map on these realized times, so that $\tau(t_k) = \mathbf{x}_{t_k}$. For a fixed sampler, we denote the resulting reverse trajectory distribution by $\tau \sim \Gamma_\theta^{\mathrm{rev}}$.

The distinction between forward random-masking states during training and reverse sampler-induced states during inference highlights the training–inference mismatch in standard MDLM training. During training, the loss is evaluated on states obtained by sampling $\mathbf{x}_0 \sim p_{\mathrm{data}}$, $t \sim \mathcal{U}(0,1)$, and $\mathbf{x}_t \sim q^{t|0}(\cdot \mid \mathbf{x}_0)$, whereas during inference the model encounters intermediate states $\tau(t_k)$ along trajectories sampled from $\Gamma_\theta^{\mathrm{rev}}$. Thus, the training state distribution need not match the inference state distribution, which can make learning less efficient.

On-Policy Distillation (OPD). OPD [1] reduces the analogous exposure bias in autoregressive models by supervising a student on prefixes sampled from its own generation process. The student first samples an on-policy sequence $\hat{\mathbf{x}} \sim p_\theta$. The teacher p_ϕ is then queried on the causal prefixes of this same sequence, producing token-level targets for the student as

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{\hat{\mathbf{x}} \sim p_\theta}\left[\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1}^{|\hat{\mathbf{x}}|} D_{\mathrm{KL}}\left(p_\phi(\cdot \mid \hat{\mathbf{x}}_{<i}) \,\|\, p_\theta(\cdot \mid \hat{\mathbf{x}}_{<i})\right)\right]. \tag{3}$$

The distinction from supervised distillation is the state distribution on which the student and teacher distributions are compared: OPD trains on prefixes induced by the student's own rollout rather than on prefixes from a fixed dataset or from teacher-generated responses.

4. On-Policy Diffusion Language Models

We propose On-Policy Diffusion Language Models (OPDLM), an on-policy method for efficiently converting an ARLM into a DLM. We begin by writing block diffusion training as a general objective over trajectory distributions, which will allow us to compare standard forward-masking training with OPDLM's reverse-trajectory training. Let Γ denote a distribution over trajectories $\tau \sim \Gamma$, viewed as maps from times to sequence states, and let $\nu(\cdot \mid \tau)$ denote a state-selection distribution over times in the trajectory. For a sampled trajectory and selected time $t \sim \nu(\cdot \mid \tau)$, we write $\mathbf{x}_t = \tau(t)$ for the selected intermediate state and $\mathbf{x}_0 = \tau(0)$ for the terminal sequence. We define the generalized block diffusion objective as

$$\mathcal{L}_{\Gamma,\nu,\phi}(\theta) = \mathbb{E}_{\tau\sim\Gamma,\, t\sim\nu(\cdot|\tau)}\left[\sum_{b=1}^{B} w(t) \sum_{i\in\mathcal{M}(\mathbf{x}_t^b)} D_{\mathrm{KL}}\left(\phi^{b,i}(\cdot \mid \mathbf{x}_0) \,\middle\|\, p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b)\right)\right]. \tag{4}$$

Here, Γ determines the source of training trajectories, ν determines which state along each trajectory is supervised, $w(t)$ is a time-dependent weight, $\phi^{b,i}$ is the target distribution at position i in block b, and $p_\theta^{b,i}$ is the DLM student's predictive distribution at the same position.

This formulation recovers the standard block-diffusion loss in Eq. (2) as a special case. Choose Γ to be the trajectory distribution induced by first sampling $\mathbf{x}_0 \sim p_{\mathrm{data}}$ and then sampling $\tau \sim \Gamma^{\mathrm{fwd}}(\cdot \mid \mathbf{x}_0)$, where $\tau(t) \sim q^{t|0}(\cdot \mid \mathbf{x}_0)$ for every t. Taking $\nu = \mathcal{U}(0,1)$, $w(t) = 1/t$ for the linear schedule $\alpha_t = 1 - t$, and hard-label targets $\phi^{b,i}(\cdot \mid \mathbf{x}_0) = \delta_{x_0^{b,i}}$ recovers Eq. (2), since $D_{\mathrm{KL}}\left(\delta_{x_0^{b,i}} \,\middle\|\, p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b)\right) = -\log p_\theta^{b,i}(x_0^{b,i} \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b)$. This trajectory-level view isolates the source of the training–inference mismatch in standard block diffusion. Training uses states induced by the data-marginal forward random-masking trajectory distribution, obtained by sampling $\mathbf{x}_0 \sim p_{\mathrm{data}}$ and then $\tau \sim \Gamma^{\mathrm{fwd}}(\cdot \mid \mathbf{x}_0)$, whereas inference uses states induced by the reverse decoding trajectory distribution $\Gamma_\theta^{\mathrm{rev}}$. In this framework, directly reducing the mismatch amounts to changing the training trajectory distribution.

Algorithm 1 OPDLM Training Step

Input: frozen ARLM teacher p_{ARLM}; student DLM p_θ; sampler $\mathcal{S}$
// **On-policy reverse rollout**
Sample reverse trajectory $\tau = (\mathbf{x}_{t_T}, \mathbf{x}_{t_{T-1}}, \ldots, \mathbf{x}_{t_0}) \sim \Gamma_\theta^{\mathrm{rev}}$
Set terminal sequence $\mathbf{x}_0 \leftarrow \tau(t_0)$
Sample non-terminal trajectory index $k \sim \mathcal{U}(\{1, \ldots, T\})$
Set training time $t \leftarrow t_k$ and on-policy state $\mathbf{x}_t \leftarrow \tau(t_k)$
Initialize loss $\mathcal{L} \leftarrow 0$
// **ARLM-supervised distillation**
for each block $b = 1, \ldots, B$ **do**
  for each masked position $i \in \mathcal{M}(\mathbf{x}_t^b)$ **do**
    Set teacher distribution $p_{\mathrm{T}} \leftarrow p_{\mathrm{ARLM}}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_0^{b,<i})$
    Set student distribution $p_{\mathrm{S}} \leftarrow p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b)$
    Update loss $\mathcal{L} \leftarrow \mathcal{L} + D_{\mathrm{KL}}(p_{\mathrm{T}} \,\|\, p_{\mathrm{S}})$
  end for
end for
Output: $\mathcal{L}$

4.1. Addressing Training–Inference Mismatch

Guided by the trajectory objective in Eq. (4), OPDLM replaces the forward masking trajectory distribution with the student's sampler-induced reverse trajectory distribution. Specifically, we sample a reverse trajectory $\tau = (\mathbf{x}_{t_T}, \mathbf{x}_{t_{T-1}}, \ldots, \mathbf{x}_{t_0}) \sim \Gamma_\theta^{\mathrm{rev}}$, where $\Gamma_\theta^{\mathrm{rev}}$ is induced by the student DLM and fixed sampler, and $t_T, \ldots, t_0$ are the realized effective denoising times along the sampled trajectory, with $t_T = 1$ and $t_0 = 0$. Here, $\mathbf{x}_0 = \tau(t_0)$ denotes the terminal generated sequence of the sampled reverse trajectory. We then sample an index $k \sim \mathcal{U}(\{1, \ldots, T\})$, set $t = t_k$, and take $\mathbf{x}_t = \tau(t_k)$. Equivalently, this samples t from the state-selection distribution $\nu_{\mathrm{rev}}(\cdot \mid \tau)$ that is uniform over the realized non-terminal times $\{t_1, \ldots, t_T\}$. Substituting this choice of Γ and ν_{rev} into Eq. (4) gives the on-policy block diffusion objective

$$\mathcal{L}_{\mathrm{on}}(\theta) = \mathbb{E}_{\tau \sim \Gamma_\theta^{\mathrm{rev}},\, t \sim \nu_{\mathrm{rev}}(\cdot \mid \tau)} \left[\sum_{b=1}^{B} w(t) \sum_{i \in \mathcal{M}(\mathbf{x}_t^b)} D_{\mathrm{KL}} \left(\phi^{b,i}(\cdot \mid \mathbf{x}_0) \,\middle\|\, p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b) \right) \right]. \tag{5}$$

This objective is on-policy because the supervised states are sampled from the same reverse decoding process used at inference. It still leaves open the choice of target distribution $\phi^{b,i}$ at each masked position. A trained DLM teacher could provide such targets, but would defeat our goal of converting an ARLM into a DLM without first training a separate DLM. We therefore define $\phi^{b,i}$ using the original frozen ARLM, and p_θ as the DLM student initialized from the same ARLM checkpoint but with blockwise-causal attention.

4.2. ARLM Supervision for On-Policy States

To define the target distribution $\phi^{b,i}$ for a partially masked diffusion state, a trained DLM teacher could be queried directly on $\mathbf{x}_t$, but OPDLM instead uses the frozen ARLM teacher. This choice serves two purposes: it avoids requiring a separately trained DLM teacher, and it transfers knowledge from the original ARLM through token-level distribution matching.

The main challenge is that the student DLM predicts from a partially masked block $\mathbf{x}_t^b$, whereas the ARLM teacher is defined over unmasked causal prefixes. We resolve this structural mismatch by using the terminal sequence $\mathbf{x}_0 = \tau(t_0)$ from the student's reverse trajectory to construct the teacher prefix. For each masked position i in block b, we define the target distribution by querying the teacher on the unmasked causal prefix:

$\phi^{b,i}(\cdot \mid \mathbf{x}_0) = p_{\text{ARLM}}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_0^{b,<i})$, where p_{ARLM} denotes the original frozen ARLM. Substituting this target distribution into Eq. (5) gives the OPDLM objective as

$$\mathcal{L}_{\text{OPDLM}}(\theta) = \mathbb{E}_{\tau \sim \Gamma_\theta^{\text{rev}}, t \sim \nu_{\text{rev}}(\cdot|\tau)} \left[\sum_{b=1}^{B} w(t) \sum_{i \in \mathcal{M}(\mathbf{x}_t^b)} D_{\text{KL}} \left(p_{\text{ARLM}}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_0^{b,<i}) \,\middle\|\, p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b) \right) \right].$$

The student is therefore trained on its own reverse-trajectory states, with supervision provided by the predictive distribution of the original ARLM. Because the target is the ARLM's full token-level predictive distribution at each supervised position, this objective encourages retention of knowledge from the original model. For the loss weighting, we follow Zhou et al. [60] and set $w(t) = 1$; although related work has explored alternative schemes [53, 15], we find this simple uniform weighting sufficient. The training procedure is visualized in Fig. 2 and detailed step-by-step in Algorithm 1.

4.3. Rollout-Length Curriculum

Although OPDLM trains on the student's own reverse trajectories, these trajectories can be unstable immediately after ARLM-to-DLM conversion. The student inherits the ARLM weights, but is now queried in a different mode: masked-token embeddings appear in the input, attention is bidirectional within each block, and predictions are made at masked positions rather than by standard next-token prediction. As a result, long early rollouts may produce low-quality terminal sequences, causing the ARLM teacher to provide targets on incoherent causal prefixes. We stabilize training using a curriculum learning inspired strategy [5, 34], progressively increasing the length of the rollout as the student improves. Let L_{min} and L_{max} denote the minimum and maximum sequence lengths used for on-policy generation. At training step s, we set

$$L_s = \min \left(L_{\text{max}}, L_{\text{min}} + \left\lfloor \frac{s}{S_{\text{warm}}} (L_{\text{max}} - L_{\text{min}}) \right\rfloor \right),$$

where S_{warm} is the number of warmup steps. Early in training, this restricts optimization to shorter rollouts, where the student learns to match the ARLM teacher on contexts primarily comprised of the prompt. As training progresses, the curriculum gradually exposes the student to longer trajectories while avoiding the cost of long, low-quality early rollouts that provide weak training signal.

5. Experiments

In this section, we present experimental results validating the benefits of On-Policy Diffusion Language Models (OPDLMs). We first outline our experimental setup (Section 5.1) and then present our main results on general-purpose benchmarks (Section 5.2). We further conduct an ablation study to disentangle the contributions of OPDLM's design choices (Section 5.3), evaluate inference-time efficiency through multi-token decoding (Section 5.4), and demonstrate OPDLM's effectiveness as a specialized DLM (Section 5.5).

5.1. Setup

Base student and teacher models. Following [60, 8], we use the Qwen3 family (0.6B to 8B) as our base ARLMs for conversion to DLMs [37]. Following our self-distillation setup, the teacher and student are same models, i.e., 4B ARLM teaches 4B student, differing only in use of logit shifting for the ARLM and their attention mask, i.e., the teacher has causal attention and the student uses block-wise causal attention.

Dataset Preparation. We train OPDLM on a curated corpus of ∼60K samples spanning four domains: math (20,222 samples), code (21,594 samples), science (10,000 samples) and chat (10,000 samples), with details in Section B.1. Since OPDLM is an on-policy method, we only keep the prompts of those datasets.

Table 1: **OPDLM 4B** and **8B** achieves competitive performance across general knowledge, mathematics, and code generation benchmarks while requiring dramatically fewer training resources - as little as **0.075B tokens**, representing up to a **15× to 7,000× reduction in training tokens** when compared to the other baselines. Cell shading in the Tokens and FLOPs rows reflects training efficiency, darker green indicates fewer tokens/FLOPs consumed.

	4B Scale		**8B Scale**				
Benchmark	SDAR-4B	**OPDLM-4B**	LLaDA-8B	Dream-7B	SDAR-8B	Fast-dLLM-v2-7B	**OPDLM-8B**
Tokens ↓	55B	**0.076B**	1500B	580B	55B	1B	**0.066B**
FLOPs (1e18) ↓	1320	**2.4**	72000	24360	2640	42	**4.2**
General Knowledge & Instruction Following							
MMLU	74.9	65.5	65.5	67.0	78.6	66.6	70.9
MMLU-Pro	50.9	46.3	37.0	43.3	56.9	41.5	53.7
GPQA-Diamond	33.0	29.1	31.8	32.1	40.2	27.3	36.1
IFEval	56.6	53.8	59.9	62.5	61.4	65.4	50.1
CEval	62.9	66.9	-	-	70.2	70.3	73.3
LiveBench	25.3	27.8	-	-	28.6	9.5	25.8
Mathematics & Reasoning							
GSM8K	89.9	87.6	78.6	81.0	91.3	83.7	87.1
MATH-500	72.8	72.8	26.6	39.2	78.6	65.6	71.2
AIME-24	10.0	14.4	2.1	0.0	10.0	10.0	14.7
AIME-25	7.5	12.6	0.4	0.0	10.0	0.0	12.4
LMB-Hard	6.9	11.1	-	-	8.9	8.9	20.0
ZebraLogic	6.3	10.5	-	-	7.8	3.5	12.9
Code Generation							
HumanEval-base	76.8	56.1	35.4	57.9	82.3	63.4	59.8
MBPP-base	80.7	57.7	31.5	68.3	79.6	63.0	48.7
LCB-v6	12.6	10.4	-	-	14.5	9.7	9.7
Codeforces	4.0	5.0	-	-	5.8	5.0	3.5

For evaluation, we report results across different categories: general knowledge, math, and coding. Unless otherwise specified, all benchmarks are evaluated under greedy static decoding [32] with block size=4. Detailed hyperparameters can be found in Section B.3.

Baselines. We compare OPDLM against four representative DLMs: **LLaDA** [32], trained from scratch with a full-attention diffusion objective; **Dream** [53], also using a full-attention diffusion objective but initialized from a pretrained AR checkpoint; and **SDAR** [8] and **Fast-dLLM-v2** [49], which adapt pretrained AR models into block diffusion models via off-policy conversion. Unless otherwise stated, all reported results use static decoding (one token per step).

5.2. OPDLM as a General-Purpose DLM

As shown in Table 1, OPDLM achieves performance competitive with strong AR-to-diffusion baselines while using two to three orders of magnitude less tokens. SDAR remains the strong baseline on several benchmarks;

Table 2: Zero-shot extended thinking in OPDLM. Although not explicitly trained to think, OPDLM retains the pre-trained ARLM prior, enabling zero-shot extended thinking at inference.

	OPDLM-4B		**OPDLM-8B**	
Benchmark	*non-think*	*think@eval*	*non-think*	*think@eval*
GSM8K	87.6	85.3	87.1	88.0
MATH-500	72.8	75.0	71.2	75.6
AIME-24	14.4	11.2	14.7	18.6
AIME-25	12.6	13.6	12.4	19.4
LMB-Hard	11.1	17.8	20.0	17.8
ZebraLogic	10.5	9.5	12.9	17.3

Table 3: Zero-shot multilingual results at 4B and 8B scales. Despite no multilingual data in training, OPDLM retains substantial multilingual ability; SDAR serves as an oracle reference.

Model	**MMMLU -lite**	**INCLUDE -lite**	**MT-AIME 2024**	**MLogiQA**
SDAR-4B	50.7	53.3	3.0	46.5
OPDLM-4B	51.6	49.6	5.3	46.5
Fast-dLLM-v2-7B	51.5	45.1	4.3	42.6
SDAR-8B	60.8	57.8	4.0	46.3
OPDLM-8B	56.0	51.9	7.9	42.0

however, this comparison is confounded by two factors: SDAR is trained on ∼55B tokens of undisclosed data, whereas OPDLM uses only ∼66M-76M tokens from the public corpus. We provide a controlled, like-for-like comparison under matched data and compute in Section 5.3, which isolates the contribution of our on-policy distillation objective from data scale and curation effects.

We highlight two additional findings. First, the comparative advantage of OPDLM correlates directly with task complexity. Although baseline models maintain an edge on saturated datasets like GSM8K, OPDLM achieves comparable or superior performance on highly rigorous benchmarks such as GPQA-Diamond, AIME, and LiveCodeBench (note that AIME and GPQA results are averaged across 32 and 8 seeds, respectively, following Cheng et al. [8]). Second, OPDLM effectively preserves the structural priors of its base ARLM. As a result, it exhibits zero-shot retention of capabilities entirely absent from the training distribution, including extended reasoning (Table 2) and multilingual proficiency (Table 3).

5.3. Ablation Study: On-Policy vs. Off-Policy Distillation

Setup. In this section, we compare off-policy distillation against OPDLM under matched data: both train on the same prompt corpus for a single data epoch, differing only in the training objective. **Off-Policy** Distillation [21] matches soft teacher distributions on offline teacher responses with random masking. We defer the comparison of SFT and OPDLM on the same data to the appendix.

OPDLM eliminates the training-inference divide of DLMs by training on states generated by the reverse diffusion process of the model itself. To achieve this, OPDLM modifies both the forward diffusion (the masking trajectory) and the reverse diffusion (the target response generation) to be entirely on-policy. To isolate the contribution of the on-policy masking trajectory, we introduce an intermediate baseline, $\textbf{OPDLM}_{\textbf{off}}$, which keeps the reverse diffusion on-policy but reverts the forward diffusion to standard random corruption.

Table 4: Ablation of training objectives under matched data and compute, at 4B and 8B scales. All variants are trained on the same prompt corpus for one data epoch. **Bold**: best in row within each scale.

	4B			8B		
Benchmark	**Off-Policy**	**OPDLM$_{\text{off}}$**	**OPDLM**	**Off-Policy**	**OPDLM$_{\text{off}}$**	**OPDLM**
MMLU	66.3	**66.7**	65.5	65.5	69.1	**70.9**
MMLU-Pro	**47.2**	47.1	46.3	51.7	52.7	**53.7**
GPQA-Diamond	**30.9**	29.9	29.1	33.6	**38.3**	36.1
IFEval	46.2	49.0	**53.8**	43.3	48.8	**50.1**
GSM8K	85.8	86.7	**87.6**	85.7	**88.1**	87.1
MATH500	73.2	**75.8**	72.8	69.2	**73.8**	71.2
AIME-24	10.9	11.2	**14.4**	**17.0**	12.9	14.7
AIME-25	11.9	**12.7**	12.6	14.3	**15.4**	12.4
HumanEval	48.8	37.8	**56.1**	54.9	54.9	**59.8**
MBPP	54.5	56.4	**57.7**	**59.5**	57.9	48.7
LCB-v6	6.4	**11.7**	10.4	6.4	**11.9**	9.7
Codeforces	2.4	4.5	**5.0**	3.7	**4.5**	3.5

OPDLM$_{\text{off}}$ ablates the impact of selecting masked states from the decoded trajectory by instead applying standard random corruption to the final generated sequence. We generate a terminal sequence $\mathbf{x}_0$ using the student's native decoding ($\mathbf{x}_0 \sim p_\theta$) and compute ARLM soft targets exactly as in OPDLM. To construct the training state, similar to block diffusion, we sample a time step $t \sim \mathcal{U}(0,1)$ and apply independent random masking $\mathbf{x}_t \sim q^{t|0}(\cdot \mid \mathbf{x}_0)$, yielding the objective:

$$\mathcal{L}_{\text{OPDLM}_{\text{off}}}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim p_\theta,\ t \sim \mathcal{U}(0,1),\ \mathbf{x}_t \sim q^{t|0}(\cdot|\mathbf{x}_0)} \left[\sum_{b=1}^{B} w(t) \sum_{i \in \mathcal{M}(\mathbf{x}_t^b)} D_{\text{KL}} \left(p_{\text{ARLM}}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_0^{b,<i}) \,\middle\|\, p_\theta^{b,i}(\cdot \mid \mathbf{x}_0^{<b}, \mathbf{x}_t^b) \right) \right] \tag{6}$$

Results. Table 4 reports the comparison. Two findings emerge. **First**, on-policy data is the primary driver of performance. Switching the training data from offline teacher generations to the model's own on-policy generations, moving from the Off-Policy baseline to OPDLM$_{\text{off}}$ while keeping the ARLM soft targets fixed, yields consistent gains across both the 4B and 8B scales. **Second**, when using on-policy data, the specific masking trajectory has a limited effect. OPDLM$_{\text{off}}$ and OPDLM effectively serve as training augmentations of one another. They differ only in whether their masked states are drawn from random corruption or the model's own decoding trajectory, yet both achieve comparable performance across benchmarks. However, we note that this study was done for a block size of 4 and leave a broader study of varying block sizes (beyond the size of 4 used in these experiments) to future work.

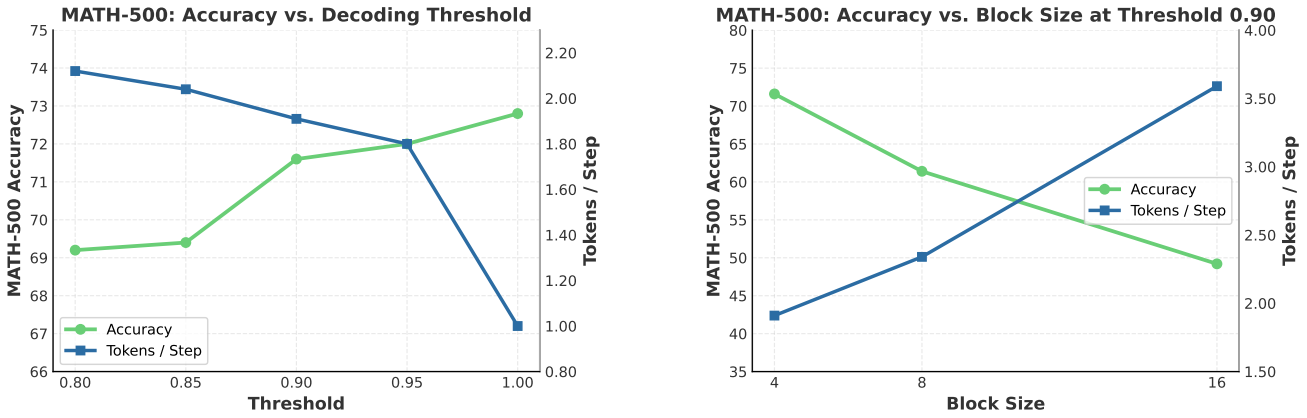

Figure 3: MATH-500 accuracy and average tokens per denoising step under different decoding configurations. **(a)** OPDLM with block size 4: decreasing the decoding threshold γ increases tokens/step, with MATH-500 accuracy as the trade-off. **(b)** At fixed threshold $\gamma = 0.9$: larger training block sizes yield more tokens/step, again at the cost of accuracy.

Table 5: OPDLM-MATH compared to TraDo [46] on math benchmarks. Even without a verifier signal or a pre-training, OPDLM-MATH achieves strong performance, particularly on the harder benchmarks. The "Thinking" variants are trained from scratch as separate models to enable extended reasoning. We also provide a reference comparison to standard DLMs.

Model	GSM8K	MATH500	AIME24
Reference			
SDAR-4B-Chat	90.2	70.2	5.0
LLaDA-8B-Instruct	82.5	37.3	0.5
Dream-7B-Instruct	72.7	38.7	0.0
SDAR-8B-Chat	91.1	74.3	11.8
4B scale			
TraDo-4B-Instruct	**91.2**	75.6	8.3
OPDLM-MATH-4B	83.8	**75.8**	**10.0**
OPDLM-MATH-4B-Thinking	**91.7**	**90.2**	**43.3**
8B scale			
TraDo-8B-Instruct	**92.3**	**78.5**	13.3
OPDLM-MATH-8B	86.2	76.6	**23.3**
TraDo-8B-Thinking	**94.2**	87.4	35.5
OPDLM-MATH-8B-Thinking	93.8	**92.4**	**50.0**

5.4. Inference Efficiency: Multi-Token Decoding

In this section, we briefly explore how OPDLM's inference throughput can be controlled through two complementary mechanisms, namely, the decoding confidence threshold and the block size used in training. More results are provided in Section A.3.

Effect of decoding threshold. For a fixed block size, OPDLM admits multiple tokens per denoising step whenever their predicted confidence exceeds a threshold γ. Section 5.3 reports this trade-off on MATH-500

with block size 4: throughput increases from 1 token/step at $\gamma = 1$ (static decoding) to over 2 tokens/step at $\gamma = 0.8$, with a moderate accuracy cost.

Effect of block size. The block size used in training OPDLM models sets an upper bound on the parallelism achievable at inference. Section 5.3 shows that at a fixed threshold $\gamma = 0.9$, increasing block size from 4 to 16 raises throughput from $\sim$2 to $\sim$3.5 tokens/step, at some cost of accuracy.

5.5. OPDLM as a Task-Specific DLM

Post-training DLMs via reinforcement learning with verifiable rewards (RLVR) [17] is a prominent approach for developing task-specialized experts [46]. However, because our framework distills knowledge directly from an ARLM teacher, we can bypass the intermediate requirement of building a general-purpose DLM. If an expert model is the primary objective, we can achieve this efficiently by training an OPDLM strictly on task-specific data.

To ensure a fair comparison in Table 5, we train OPDLM on the same data used by TraDo [46]: 8K level 3-5 hard problems from the MATH training set [20]. The key difference lies in the starting point: TraDo initializes from an SDAR-pretrained DLM and applies RL with verifiable rewards, whereas OPDLM starts directly from the AR Qwen3 checkpoint and applies on-policy distillation with no rewards or pretrained DLM stage. We train two variants: **OPDLM-MATH**, which distills the teacher in non-thinking mode, and **OPDLM-MATH-Thinking**, which distills the teacher with its thinking behavior enabled.

6. Conclusion

We introduce On-Policy Diffusion Language Models (OPDLM), for converting autoregressive language models (ARLMs) into diffusion language models (DLMs). Existing ARLM-to-DLM conversion methods face two challenges: retaining knowledge from the original ARLM after changing the training objective and attention pattern, and reducing the training–inference state mismatch between forward random masking during training and reverse unmasking at inference. OPDLM addresses these challenges by adapting on-policy distillation to DLM conversion. The student DLM is initialized from the ARLM, samples its own reverse diffusion trajectories, and is supervised on those trajectories using token-level distributions from the original frozen ARLM teacher. Empirically, OPDLM requires **15**$\times$ to **7,000**$\times$ fewer training tokens than prior DLM baselines while achieving competitive performance across a broad range of tasks. These results reframe ARLM-to-DLM conversion as an efficient post-training procedure, opening a practical path for adapting future ARLMs into diffusion-based language models.

Acknowledgments

This work was supported in part by the National Institutes of Health (NIH) under grant U01AG070112 and by the Advanced Research Projects Agency for Health (ARPA-H) under grant 1AY1AX000053.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

[2] 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. URL https://arxiv.org/abs/2503.09573.

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

[4] 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*, 2021. URL https://arxiv.org/abs/2108.07732.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

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

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

[8] Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025. URL https://arxiv.org/abs/2510.06303.

[9] 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.

[10] DeepSeek-AI. Deepseek-v4 technical report. Technical report, DeepSeek-AI, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

[11] Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvalddson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[13] Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. *arXiv preprint arXiv:2512.15489*, 2025. URL https://arxiv.org/abs/2512.15489.

[14] Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael P. Brenner. HARDMath: A benchmark dataset for challenging problems in applied mathematics. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=nDTvP6tBMd.

[15] 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*, 2026. URL https://arxiv.org/abs/2512.14067.

[16] Zitian Gao, Haoming Luo, Lynx Chen, Jason Klein Liu, Ran Tao, Joey Zhou, and Bryan Dai. What makes diffusion language models super data learners? *arXiv*, 2025. doi: 10.48550/arxiv.2510.04071.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[18] Tomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *bioRxiv*, pages 2024–07, 2024.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[20] 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 preprint arXiv:2103.03874*, 2021.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 2015. doi: 10.48550/arxiv.1503.02531.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[23] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[24] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/c6ec1844bec96d6d32ae95ae694e23d8-Abstract-Datasets_and_Benchmarks.html.

[25] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[26] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

[27] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[28] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

[29] Bill Yuchen Lin, Ronan Le Bras, and Yejin Choi. *ZebraLogic: Benchmarking the Logical Reasoning Ability of Language Models*, 2024. URL https://huggingface.co/spaces/allenai/ZebraLogic.

[30] Anant Mehta, Xiyuan Wei, Xingyu Chen, and Tianbao Yang. Breaking the limits of open-weight clip: An optimization framework for self-supervised fine-tuning of clip. *arXiv preprint arXiv:2601.09859*, 2026.

[31] Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv*, 2025. doi: 10.48550/arxiv.2511.03276.

[32] 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

[33] NVIDIA, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, et al. NVIDIA nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025. URL https://arxiv.org/abs/2508.14444.

[34] Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KJvHnl3kUv.

[35] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[36] Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces. https://huggingface.co/datasets/open-r1/codeforces, 2025.

[37] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[38] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

[39] Angelika Romanou, Negar Foroutan, Anna Sotnikova, Zeming Chen, Sree Harsha Nelaturu, Shivalika Singh, Rishabh Maheshwary, Micol Altomare, Mohamed A Haggag, Alfonso Amayuelas, et al. Include: Evaluating multilingual language understanding with regional knowledge. *arXiv preprint arXiv:2411.19799*, 2024.

[40] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

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

[42] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026. URL https://arxiv.org/abs/2601.19897.

[43] Guijin Son, Jiwoo Hong, Hyunwoo Ko, and James Thorne. Linguistic generalizability of test-time scaling in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14333–14368, 2025.

[44] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[45] Yuchuan Tian, Yuchen Liang, Jiacheng Sun, Shuo Zhang, Guangwen Yang, Yingte Shu, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, Hanting Chen, Xinghao Chen, and Yunhe Wang. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*, 2026. URL https://arxiv.org/abs/2512.06776.

[46] 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*, 2025.

[47] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html.

[48] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

[49] 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*, 2025.

[50] Yong Xie, Karan Aggarwal, and Aitzaz Ahmad. Efficient continual pre-training for building domain specific large language models. *Findings of the Association for Computational Linguistics ACL 2024*, pages 10184–10201, 2024. doi: 10.18653/v1/2024.findings-acl.606.

[51] Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.365/.

[52] 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*, 2025.

[53] 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. URL https://arxiv.org/abs/2508.15487.

[54] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=2a36EMSSTp.

[55] Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: Acing coder rl via automated test-case synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12023–12040, 2025. URL https://aclanthology.org/2025.acl-long.587/.

[56] Yidan Zhang, Yu Wan, Boyi Deng, Baosong Yang, Hao-Ran Wei, Fei Huang, Bowen Yu, Dayiheng Liu, Junyang Lin, and Jingren Zhou. P-mmeval: A parallel multilingual multitask benchmark for consistent evaluation of llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4809–4836, 2025.

[57] Yifan Zhang and Math-AI Team. *American Invitational Mathematics Examination (AIME) 2024*, 2024. URL https://huggingface.co/datasets/math-ai/aime24.

[58] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

[59] 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 preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

[60] Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. *arXiv preprint arXiv:2602.22661*, 2026.

[61] Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *arXiv preprint arXiv:2409.19681*, 2024.

Data-Efficient Autoregressive-to-Diffusion Language Models via On-Policy Distillation

Appendix

A. Additional Experimental Results

A.1. Results at Smaller Scales

Table 6: Main results across 0.6B and 1.7B scales.

	0.6B Scale		1.7B Scale		
Benchmark	**Simple-dLLM**	**OPDLM-0.6B**	**SDAR-1.7B**	**Fast-dLLM-v2-1.5B**	**OPDLM-1.7B**
Tokens ↓	46B	**0.048B**	55B	1B	**0.072B**
FLOPs (1e18) ↓	165.6	**0.23**	561.0	9.0	**0.98**
MMLU	39.1	42.0	62.9	55.1	53.8
MMLU-Pro	13.8	20.4	37.0	24.1	31.1
GPQA-Diamond	22.2	24.0	29.8	24.8	26.8
IFEval	35.5	24.8	43.4	47.0	39.7
GSM8K	46.3	42.1	80.1	62.0	71.0
MATH500	15.8	28.0	63.2	39.4	53.0
HumanEval-Base	45.7	20.1	59.8	43.9	35.4
MBPP-Base	54.0	24.9	68.5	50.0	45.0

Table 6 reports OPDLM at 0.6B and 1.7B scales against Simple-dLLM [60], Fast-dLLM-v2 [49], and SDAR [8]. The trends mirror our main results: OPDLM matches or surpasses comparable-scale baselines on math and reasoning benchmarks while using significantly fewer tokens, and underperforms on coding benchmarks where our data corpus has limited coverage.

A.2. Effect of Teacher Model Size

Table 7: Effect of teacher size on OPDLM-MATH-4B and OPDLM-MATH-8B. Same-size teachers (self-distillation) match or outperform a larger Qwen-32B teacher.

Model	**GSM8K**	**MATH500**	**AIME24**
Qwen-4B (Non-thinking)	90.2	84.8	25.0
Qwen-8B (Non-thinking)	91.4	87.4	29.1
Qwen-32B (Non-thinking)	93.6	88.6	31.0
OPDLM-MATH-4B (from Qwen-4B)	83.8	75.8	10.0
OPDLM-MATH-4B (from Qwen-32B)	84.9	72.2	10.0
OPDLM-MATH-8B (from Qwen-8B)	86.2	76.6	23.3
OPDLM-MATH-8B (from Qwen-32B)	86.4	71.0	10.0

Table 8: Effect of teacher size on OPDLM-0.6B. A larger Qwen3-4B teacher outperforms self-distillation, in contrast to the results in Table 7.

Benchmark	**from Qwen3-0.6B** *(self-distillation)*	**from Qwen3-4B**
MMLU	42.0	42.5
MMLU-Pro	20.4	19.3
GPQA-Diamond	24.0	18.2
IFEval	24.8	21.4
GSM8K	42.1	47.9
MATH500	28.0	36.2
HumanEval-Base	20.1	23.8
MBPP-Base	24.9	25.4

In our main experiments, OPDLM employs self-distillation: a 4B student is distilled from a 4B teacher, and an 8B student from an 8B teacher. A natural question is whether the optimal teacher size depends on the student scale.

Table 7 and Table 8 reveal an interesting finding. For larger students (4B and 8B), self-distillation matches or outperforms a stronger Qwen-32B teacher. For smaller students (0.6B), self-distillation is no longer sufficient, and a larger Qwen3-4B teacher provides some gains. We hypothesize that a 0.6B teacher does not provide rich enough predictive signal on student-generated rollouts. Together, these results suggest the optimal teacher–student configuration depends on student size: same-size teachers suffice for capable students (4B+), while smaller students benefit from moderately larger teachers. A systematic study of this scaling behavior is left to future work.

A.3. Additional Results for Multi-token Decoding

We extend the analysis in Section 5.4 along two complementary axes: block size and decoding confidence threshold. Fig. 4 sweeps block size at four fixed thresholds, while Fig. 5 sweeps the threshold at three fixed block sizes. Both figures evaluate three benchmarks (MATH-500, GPQA-Diamond, MBPP) to verify that the trade-off generalizes beyond mathematical reasoning.

Across all configurations, the same accuracy-throughput trade-off holds: larger block sizes and lower thresholds yield more tokens per denoising step at the cost of accuracy.

Fig. 6 reports the average number of tokens generated per denoising step over the course of training. Throughput rises rapidly and stabilizes thereafter, suggesting that the model converges quickly to its block-size-determined parallelism level. The relative ordering matches what we observe at inference time: block size 16 achieves the highest throughput throughout training, followed by block size 8 and block size 4, with block size 16 also showing the largest variance during the early phase.

A.4. Multi-Seed Evaluation on GPQA-Diamond, AIME-24, AIME-25

Table 9 reports OPDLM accuracy with standard deviations on GPQA-Diamond, AIME-24, and AIME-25. As these benchmarks are small and single-run scores can fluctuate, we evaluate each model over multiple random seeds and report mean $\pm$ std.

Table 9: Performance with std on GPQA-Diamond, AIME-24, AIME-25. We evaluate over N random seeds and report mean $\pm$ std.

Model	GPQA-Diamond (N=8)	AIME-24 (N=32)	AIME-25 (N=32)
OPDLM-0.6B	24.0 ± 2.3	-	-
OPDLM-1.7B	26.8 ± 1.3	-	-
OPDLM-4B	29.1 ± 2.7	14.4 ± 3.7	12.6 ± 3.6
OPDLM-4B think@eval	-	11.2 ± 4.4	13.6 ± 5.0
OPDLM-8B	36.1 ± 2.2	14.7 ± 3.4	12.4 ± 5.0
OPDLM-8B think@eval	-	18.6 ± 4.2	19.4 ± 4.2

A.5. Comparison with Supervised Fine-Tuning

In this sub-section, we extend the off-policy ablation to also measure the impact of training with the BD3LM loss directly on our ARLM-generated data. We generate one response per prompt and apply the BD3LM loss [2], training for one epoch with block size 4.

Table 10: Comparison of SFT and OPDLM under matched data and compute at the 4B scale. Both variants are trained on the same prompt corpus for one data epoch. **Bold**: best in column.

Method	MMLU	MMLU-Pro	GPQA	IFEval	GSM8K	MATH500	AIME-24	AIME-25	HumanEval	MBPP
SFT	**66.6**	**48.4**	**29.7**	44.7	86.8	**75.4**	12.4	10.4	**65.9**	**58.7**
OPDLM	65.5	46.3	29.1	**53.8**	**87.6**	72.8	**14.4**	**12.6**	56.1	57.7

Table 11: Effect of dynamic sampling on OPDLM and SFT at the 4B scale. We report accuracy under static decoding (one token per step) and under dynamic sampling (confidence threshold=0.9). Δ is the per-benchmark change under dynamic sampling. We also report the std for AIME benchmarks.

Method	Decoding	GSM8K	MATH500	AIME-24	AIME-25	HumanEval	MBPP
OPDLM	Static	87.6	72.8	14.4 $\pm$3.7	12.6 $\pm$3.6	53.0	58.7
	Dynamic	86.0	71.6	12.5 $\pm$3.6	14.7 $\pm$4.2	53.0	56.1
	Δ	-1.6	-1.2	-1.9	$+2.1$	0.0	-2.6
SFT	Static	86.8	75.4	12.4 $\pm$3.6	10.4 $\pm$3.1	65.9	59.0
	Dynamic	83.9	71.2	13.4 $\pm$4.5	9.6 $\pm$2.2	59.8	58.5
	Δ	-2.9	-4.2	$+1.0$	-0.8	-6.1	-0.5

We first observe that SFT on responses generated by the original ARLM is a simple and efficient recipe, attaining performance comparable to OPDLM across benchmarks at the 4B scale using a block size of 4 (Table 10). We view this as a useful and previously underexplored insight; i.e., the setting in which an ARLM produces the exact data later used to convert it into a DLM has rarely been studied, largely because pre-training corpora are seldom released (with the exception of Zhou et al. [61], which instead collects data from various data sources).

Despite their comparable performance on standard benchmarks, SFT and OPDLM diverge significantly under dynamic-sampling-based decoding (Table 11). OPDLM is trained on its own generations, with the data generated using dynamic sampling. As a result, its accuracy under dynamic sampling is highly robust, close to that of one-token-per-step decoding. In contrast, the SFT model is trained on offline responses under

random masking and exhibits a much larger average performance drop under dynamic sampling. This drop highlights the train-inference divide native to standard DLMs, a limitation that SFT fails to address. Together, these results suggest that the two approaches occupy complementary regimes: SFT on ARLM responses offers a strong, efficient baseline, while OPDLM provides superior robustness across complex inference-time samplers.

B. Experiment Details

B.1. Pretraining Datasets Details

Table 12: Pretraining dataset details.

Domain	#Samples	Sources
Math	20,222	DAPO, Nemotron-v2-Math
Code	21,594	TACO, KodCode-Light-RL, AceCode
Science	10,000	Nemotron-v2-STEM
Chat	10,000	Nemotron-v2-Chat
Total	**61,816**	-

Our pretraining corpus of 62K samples is constructed from a mixture of domains. For mathemtical reasoning, we use 22k samples from DAPO [54] and Nemotron-v2-Math [13]. 20k samples of coding data was collated from TACO [28], KodCode-Light-RL [51], and AceCode [55]. Finally, from Nemotron-v2 [33] we sample 10k examples of STEM and 10k examples of Chat.

B.2. Evaluation Benchmark Details

For evaluation dataset, we use the following datasets

- **General Knowledge & Instruction Following**: MMLU [19], MMLU-Pro [47], GPQA-Diamond [38], IFEval [59], CEval [24], LiveBench [48].
- **Mathematics & Reasoning**: GSM8K [9] , MATH-500 [19], AIME-24 [57], AIME-25 [11], LMB-Hard [14], ZebraLogic [29].
- **Code Generation**: HumanEval [7], MBPP [4], LiveCodeBench-v6 [26], Codeforces [36].
- **Multilingual**: MMMLU-lite [47], INCLUDE-lite [39], MT-AIME2024 [43], MLogiQA [56].

For evaluation, we apply the official Qwen3 chat template [52]. For mathematics benchmarks, we append the instruction "`Please reason step by step, and put your final answer within \boxed{}.`" to each prompt. For multiple-choice benchmarks, we use "`Please show your choice in the answer field with only the choice letter, e.g., \"answer\": \"C\".`"

B.3. Hyperparameters

Table 13 lists the hyperparameters for general-purpose OPDLM. For OPDLM-MATH (Section 5.5), we override only the parameters in Table 14; non-thinking and thinking variants differ only in rollout length.

B.4. FLOPs Calculation

We follow the following FLOP estimation [27]: a forward pass through a model with N parameters costs approximately $2N$ FLOPs per token, and a forward + backward pass costs approximately $6N$ FLOPs per token. In OPDLM, the teacher is queried in inference mode (forward only) and the student is updated via gradient descent (forward + backward), so the total training FLOPs are estimated as

Table 13: Hyperparameter settings for General-Purpose OPDLM training.

Hyperparameter	Value
Number of data epochs	1
Tasks per rollout	128
Effective training batch size	8
Learning rate	1×10^{-5}
Final learning rate	1×10^{-6}
Learning rate schedule	Cosine with warmup
Warmup steps	5
Rollout block size	4
Rollout temperature	1.0
Initial maximum rollout length	100
Final maximum rollout length	4,000
Rollouts to reach final generation length	10
Remasking strategy	Dynamic
Dynamic remasking confidence threshold	0.9
Divergence	Forward KL

Table 14: Hyperparameter overrides for OPDLM-MATH training. All other hyperparameters follow Table 13.

Hyperparameter	Value
Number of data epochs	10
Final maximum rollout length (non-thinking/thinking)	2,000/8,000
Rollouts to reach final generation length (non-thinking/thinking)	10/20

$$\text{FLOPs} \approx 2N_{\text{teacher}} \cdot T + 6N_{\text{student}} \cdot T, \tag{7}$$

where T is the total number of training tokens, computed as the number of training samples times their generation length. Generation lengths follow a curriculum schedule (Table 13 and Table 14), and T is summed over all rollout stages. We note that the student forward pass during training operates in block-diffusion mode at roughly 1.5 tokens/step rather than 1 token/step; the formula above conservatively assumes one student forward per token, slightly overestimating the FLOPs charged to OPDLM.

B.5. Compute Usage

Compute Usage. Final model checkpoints were trained on H200 GPUs, while ablations and intermediate experiments were conducted on A6000 and A100 GPUs. Notably, our training pipeline is lightweight enough that all models below 4B parameters can be trained on as few as 2 A6000 GPUs.

C. Limitations, Future Directions and Broader Impacts

Limitations Although OPDLM achieves competitive performance on a wide range of benchmarks, there is still room for improvement on certain tasks compared to the strongest baselines. We believe that improving data quality is a key direction for further performance gains. However, the pretraining datasets used by most

baselines are not publicly available, making controlled comparisons difficult. Furthermore, while we explore thinking-mode distillation in the specialized math setting, expanding this reasoning distillation to broader, more general datasets remains an area for future work.

Future Directions

- **Data preparation.** Our experiments use a $\sim$60K-prompt corpus assembled from publicly available sources. Building a high quality and more diverse training corpora is a promising direction. Given OPDLM's strong data efficiency, investments in data curation are likely to introduce substantial performance gains.
- **Reasoning distillation at scale.** We explored thinking-mode distillation only in the specialized math setting (Section 5.5), where it substantially boosts performance on hard reasoning benchmarks (e.g., AIME24 reaching 50% at 8B). Extending chain-of-thought distillation to general-domain training would test whether OPDLM can transfer richer reasoning behaviors beyond mathematics, and is a natural next step toward a fully general thinking-enabled diffusion LLM.
- **Cross-size and cross-family distillation.** All main experiments use self-distillation between same-size, same-family teacher and student models, which may limit quality of the supervision signal. A preliminary ablation (Section A.2) suggests that naively switching to larger teachers may not automatically help, motivating a more careful study of how teacher–student model size mismatches affect the final performances. Distillation across model families (e.g., Llama and Qwen models) is also worth investigating.

Broader Impact Our work studies the efficient conversion of autoregressive language models (ARLMs) into diffusion language models (DLMs), and carries implications across both efficiency and societal dimensions. First, OPDLM substantially reduces the training cost of obtaining a DLM, requiring up to 7000$\times$ fewer training tokens than from-scratch DLM pre-training. This directly lowers the energy footprint of producing competitive generative models, contributing toward more sustainable development of large-scale AI systems. Second, ARLMs decode one token per step, which becomes prohibitive for long-form generation. By framing ARM-to-DLM conversion as a post-training procedure, OPDLM offers a natural path to faster inference through parallel multi-token decoding, further reducing the energy cost of deployment at scale. Finally, OPDLMs inherit the broader risks associated with generative AI systems, including the potential to reflect or amplify biases present in pre-training and post-training data. We encourage practitioners deploying OPDLMs to apply standard mitigations such as bias auditing, content filtering, and downstream alignment.

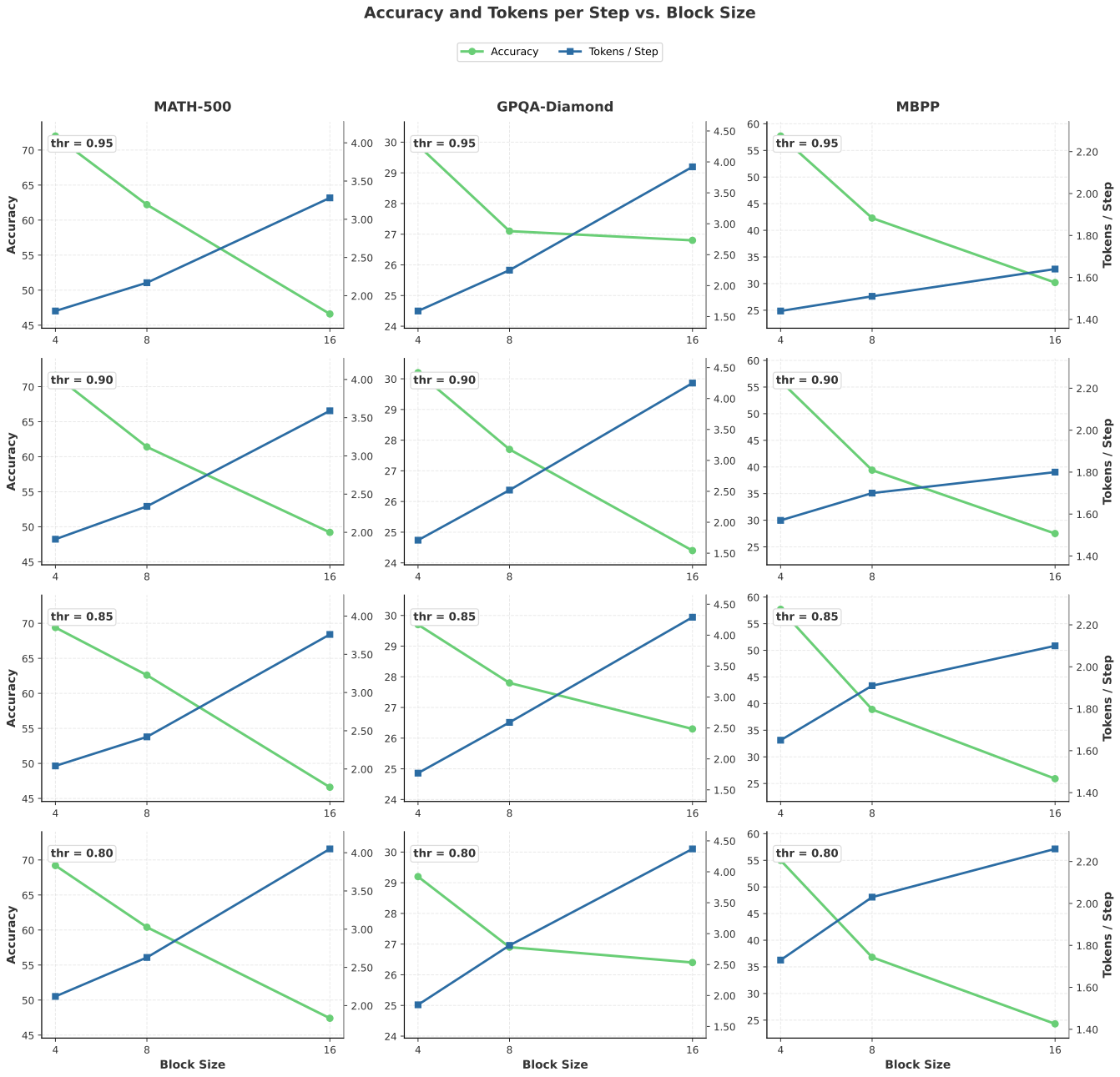

Figure 4: Effect of block size at different decoding thresholds across MATH-500, GPQA-Diamond, and MBPP. Each row corresponds to a fixed threshold $\gamma \in \{0.95, 0.90, 0.85, 0.80\}$.

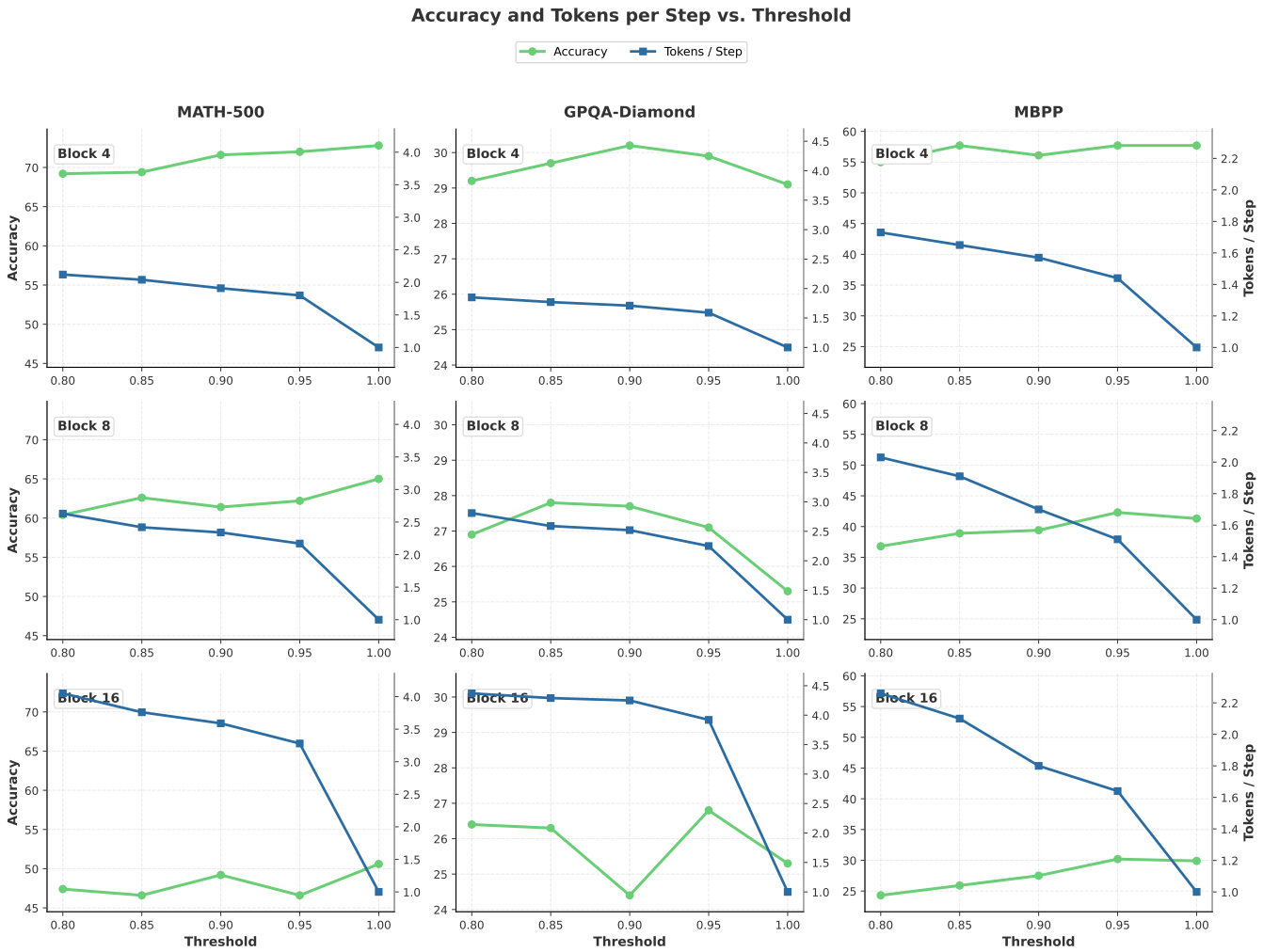

Figure 5: Effect of decoding threshold at different block sizes across MATH-500, GPQA-Diamond, and MBPP. Each row corresponds to a fixed block size $\in \{4, 8, 16\}$.

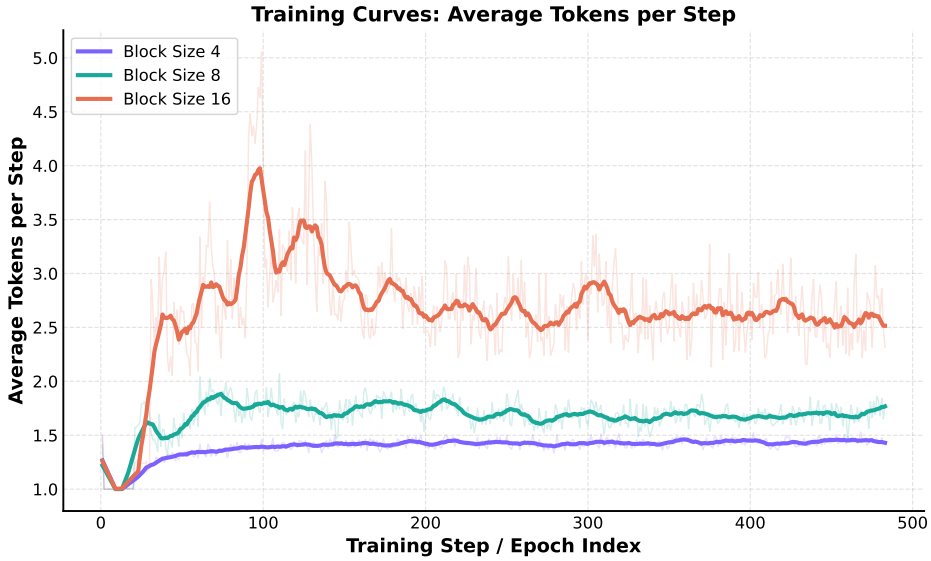

Figure 6: Training curves showing the average number of tokens generated per denoising step across block sizes. Larger block sizes produce more tokens per step, with block size 16 exhibiting the highest average token throughput during training.