

Fast Block Diffusion Training via Multi-Mask Context Sharing

Tengyu Pan¹, Weilin Zhao¹, Yutao Sun¹, Yanghao Li^{1*},
Jianyong Wang^{1,3*}, Xu Han^{1,2,3}

¹Department of Computer Science and Technology, Tsinghua University, Beijing.

²Beijing National Research Center for Information Science and Technology.

³Jiangsu Collaborative Innovation Center for Language Ability,
Jiangsu Normal University, Xuzhou.
pty23@mails.tsinghua.edu.cn

Abstract

Block Diffusion Language Models generate multiple blocks of text sequentially, while tokens within each block are denoised in parallel, enabling efficient parallel decoding. However, their training suffers from low computational utilization: each sample is duplicated into a clean context and a masked copy, yet only the masked portion contributes to the training loss, resulting in merely 50% compute utilization. In this paper, we propose **Multi-Mask Training (MMT)**, a simple yet effective training paradigm that significantly improves computational efficiency through context sharing. Our key idea is to generate K different mask patterns for the same clean text and process them together in a single forward pass via a sparse attention mechanism, where each masked segment attends to the shared clean context while remaining isolated from other segments. In pretraining, our proposed approach yields better performance under matched compute, despite being exposed to fewer unique training instances. Under a matched-step protocol, MMT maintains comparable downstream quality while improving training efficiency in AR-to-Diffusion adaptation. This reveals a compelling trade-off between data uniqueness and training efficiency in block diffusion models, demonstrating that the rich training signals from diverse mask samplings can effectively compensate for reduced exposure to unique data.

1 Introduction

The success of Large Language Models (Brown et al., 2020; Bommasani et al., 2021; Han et al., 2021; OpenAI, 2023) has been largely driven by autoregressive (AR) architectures. However, their sequential, token-by-token generation paradigm imposes significant latency bottlenecks during inference. Recently, Diffusion Language Models (Nie

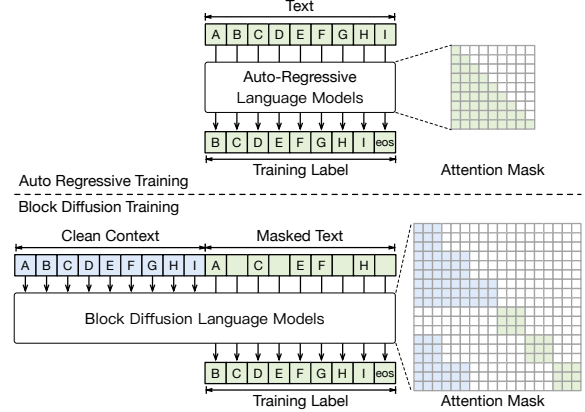


Figure 1: The training pattern of Auto-Regressive Language Models and Block Diffusion Language Models.

et al., 2025b; Khanna et al., 2025; Song et al., 2025; Google DeepMind, 2025) have emerged as promising alternatives. By training the model to generate tokens in parallel, these models offer a flexible trade-off between generation quality and inference speed.

Block Diffusion Language Models (Arriola et al., 2025; Wu et al., 2025a; Cheng et al., 2026; Tian et al., 2025; Bie et al., 2025) further bridge the gap between autoregressive and diffusion paradigms. These models decompose the generation process into sequential blocks, where tokens within each block are generated via diffusion while blocks themselves are produced autoregressively. This hybrid formulation enables variable-length generation, supports KV caching across blocks, and allows parallel token sampling within each block, making it a compelling architecture for practical deployment.

Despite their promising inference properties, Block Diffusion Language Models suffer from a critical training inefficiency. As illustrated in Figure 1, each training sample is duplicated into two copies: one serves as the clean context, while the other is masked and used for loss computation.

* indicates corresponding authors.

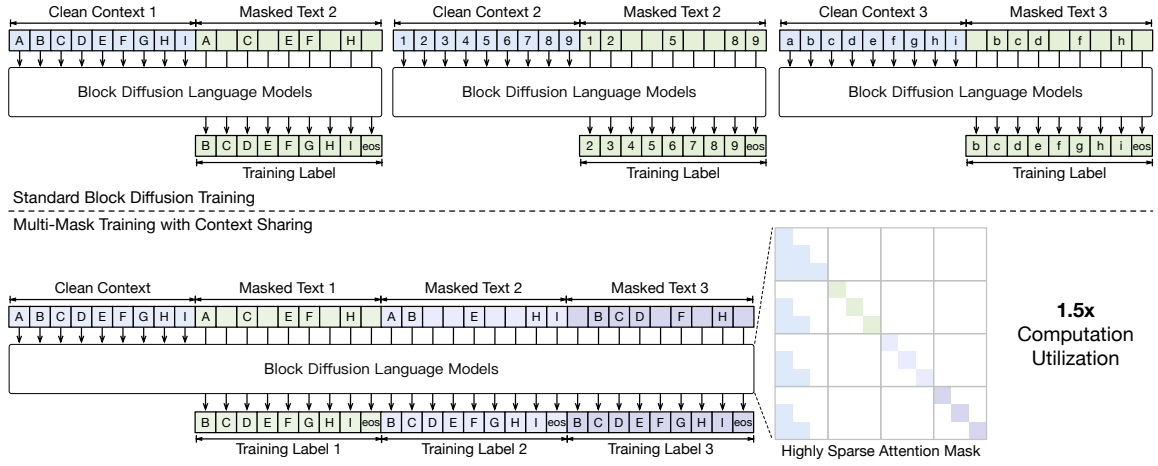


Figure 2: The training pattern of MMT compared to the original training pattern of Block Diffusion Language Models, when the number of different mask patterns K is 3. MMT trains on fewer training instances but instead trains on more mask patterns.

This duplication results in a mere 50% computational utilization, as half of the forward pass is spent on the context that does not contribute to the training signal. In contrast, autoregressive models achieve near-100% utilization since every token contributes to the training loss. Beyond the wasted FLOPs, this inefficiency also incurs substantial memory and I/O overhead: the clean context occupies precious GPU memory during forward and backward passes, yet contributes zero gradients to the model update.

In this paper, we propose **Multi-Mask Training with context sharing (MMT)**, a simple yet effective training paradigm that significantly improves computational utilization through a carefully designed attention mechanism. Our key insight is that instead of processing one mask pattern per sample, we can generate K different mask patterns for the same clean text and process them together in a single forward pass. As illustrated in Figure 2, by concatenating the shared clean prefix with K masked suffixes and employing a *context-sharing attention mask*, each masked segment attends to the clean context while remaining isolated from other segments. The resulting attention mask is highly sparse and can be efficiently implemented with FlexAttention (Dong et al., 2024). This design amortizes the cost of the clean context across K independently masked contexts, improving the computational utilization from 50% to $\frac{K}{1+K}$. Such efficiency gain comes with a natural trade-off: under a fixed training compute budget, processing K masked views jointly means seeing fewer *unique* samples, which could potentially hurt model qual-

ity due to reduced data diversity. A central question we investigate in this paper is therefore whether the efficiency gain brought by context sharing can **outweigh** this loss of sample diversity. Our pretraining experiments show that MMT achieves better model quality than the standard Block Diffusion training paradigm under matched compute, confirming that the efficiency gain not only compensates for but ultimately surpasses the cost of reduced data diversity. Our AR-to-Diffusion adaptation experiments instead follow a matched-step protocol, under which $K = 2$ achieves comparable downstream performance while requiring substantially less wall-clock training time.

Our contributions are summarized as follows:

- We identify the low computational utilization in Block Diffusion training, where the clean context incurs computation but does not contribute to the loss.
- We propose MMT, a multi-mask training paradigm with context sharing that processes multiple masked views of the same sample in a single forward pass via a specially designed sparse attention pattern.
- Experiments validate the effectiveness of MMT in both pretraining and AR-to-Diffusion adaptation scenarios. In pretraining, our proposed method achieves consistent loss improvements under matched training compute. In AR-to-Diffusion adaptation, MMT reaches the same generation quality with significantly less training time.

2 Related Work

2.1 Diffusion Language Models

Diffusion models have achieved remarkable success in continuous domains such as image and audio generation (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Peebles and Xie, 2023). Recently, there has been growing interest in adapting diffusion-based methods to discrete text generation. Mercury (Khanna et al., 2025), Seed Diffusion (Song et al., 2025), and Gemini Diffusion (Google DeepMind, 2025) are industry-developed diffusion language models that demonstrate ultra-fast inference capabilities. In the open-source community, early methods (Li et al., 2022; Lovelace et al., 2023; Gong et al., 2023; Han et al., 2023; Strudel et al., 2022; Chen et al., 2023) apply continuous-space diffusion directly to language, while more recent methods (Hoogeboom et al., 2021, 2022; He et al., 2023; Lou et al., 2024; Ou et al., 2025; Sahoo et al., 2024) leverage discrete diffusion to achieve better scalability and performance. In this paper, we mainly focus on the discrete version, also known as **masked diffusion language models**. LLaDA (Nie et al., 2025b) trains an 8B-scale masked diffusion language model from scratch, demonstrating competitive performance with autoregressive counterparts. DREAM (Ye et al., 2025) adapts pre-trained autoregressive models into diffusion language models through continued training. These methods offer a promising alternative to autoregressive generation by enabling parallel token sampling, but they typically generate sequences of fixed length and lack the flexibility of variable-length generation.

2.2 Block Diffusion Language Models

Block Diffusion Language Models bridge the gap between autoregressive and diffusion paradigms by decomposing generation into sequential blocks. Ma et al. (2025); Wu et al. (2025b) use approximate KV caching to make a diffusion language model generate block-by-block. BD3-LMs (Ariola et al., 2025) trains a native block diffusion language model from scratch. Fast-DLLM v2 (Wu et al., 2025a) and SDAR (Cheng et al., 2026) adapt pre-trained autoregressive models into block diffusion models, enabling efficient conversion of existing AR checkpoints. LLaDA 2.0 (Bie et al., 2025) scales block diffusion language models up to 100B parameters, demonstrating the scalability of this approach. These models support variable-length

generation, enable KV caching across blocks, and allow parallel token sampling within each block. However, their training suffers from low computational utilization due to the context duplication, which our method addresses.

2.3 Data Efficiency of Diffusion Language Models

Recent studies (Prabhudesai et al., 2025; Ni et al., 2025a,b) have revealed that diffusion language models are “super data learners” that can extract more information from the same amount of training data through diverse mask patterns. Another related approach is *complementary masking* (Wu et al., 2025a), which uses an inverted mask paired with a standard mask to enhance data utilization. However, these methods suffer from efficiency drawbacks. The former relies on implicit accumulation of diversity over repeated epochs with single-mask sampling, while the latter doubles the batch size and processes each corrupted sequence independently, failing to exploit the shared clean context. In contrast, our method transforms this implicit benefit into an explicit, efficient training strategy. By processing multiple mask patterns simultaneously with **context sharing**, we actively enforce diverse supervision within each step while eliminating the computational waste of encoding the same context repeatedly.

3 Method

3.1 Preliminaries

Notation. Let $\mathbf{x} = (x_1, \dots, x_N)$ denote a sequence of N tokens, $\mathbf{M}$ the special mask token, and θ the parameters of the language model. Autoregressive and diffusion language models share the same Transformer architecture, differing only in their input representations and training objectives.

Autoregressive Language Models. Autoregressive (AR) language models are trained to minimize the negative log-likelihood of the sequence:

$$\mathcal{L}_{\text{AR}} = -\mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\sum_{i=1}^N \log p_{\theta}(x_i | x_{<i}) \right]. \quad (1)$$

Masked Diffusion Language Models. Masked diffusion language models (MDLMs) (Nie et al., 2025b) define a discrete diffusion process over token sequences. Let $t \sim \mathcal{U}(0, 1)$ denote a noise level, and let $\mathbf{x}^t$ denote the corrupted sequence obtained by independently masking each token x_i

with probability t . Formally, for each position i :

$$x_i^t = \begin{cases} \text{M} & \text{with probability } t, \\ x_i & \text{otherwise.} \end{cases} \quad (2)$$

The model is trained to predict the original tokens from the corrupted sequence:

$$\begin{aligned} \mathcal{L}_{\text{MDLM}} &= -\mathbb{E}_{t, \mathbf{x}, \mathbf{x}^t} \left[\frac{1}{t} \sum_{i=1}^N \mathbb{I}[x_i^t = \text{M}] \ell_{\theta, i}^{\text{MDLM}}(t) \right], \\ \ell_{\theta, i}^{\text{MDLM}}(t) &= \log p_{\theta}(x_i | \mathbf{x}^t). \end{aligned} \quad (3)$$

Block Diffusion Language Models. Block diffusion language models (BDLMs) (Arriola et al., 2025) extend MDLMs by partitioning the sequence into contiguous blocks of size B . The sequence is divided into $N_b = N/B$ blocks, $\mathbf{x} = (\mathbf{b}_1, \dots, \mathbf{b}_{N_b})$. During training, each block $r \in \{1, \dots, N_b\}$ is assigned an independently sampled noise level $t_r \sim \mathcal{U}(0, 1)$. Tokens within $\mathbf{b}_r$ are then masked independently with probability t_r , while the causal structure across blocks is maintained. For each token x_i , let $j = \lceil i/B \rceil$ denote its block index in the following objective:

$$\begin{aligned} \mathcal{L}_{\text{BDLM}} &= -\mathbb{E}_{t, \mathbf{x}, \mathbf{x}^t} \left[\sum_{i=1}^N \frac{\mathbb{I}[x_i^t = \text{M}]}{t_j} \ell_{\theta, i}^{\text{BDLM}}(t_j) \right], \\ \ell_{\theta, i}^{\text{BDLM}}(t_j) &= \log p_{\theta}(x_i | \mathbf{b}_{< j}, \mathbf{b}_j^{t_j}). \end{aligned} \quad (4)$$

where $\mathbf{b}_{< j}$ denotes the clean preceding blocks in $\mathbf{x}$, and $\mathbf{b}_j^{t_j}$ denotes the noisy version of the j -th block in $\mathbf{x}^t$. The specialized attention mask used for BDLM training is illustrated in Figure 1.

3.2 MMT: Multi-Mask Training

Training formulation. Given a clean sequence $\mathbf{x}$, MMT shares this clean sequence as the common context and generates K corrupted sequences for denoising training. For each view $k \in \{1, \dots, K\}$, we independently sample block-wise noise levels $\mathbf{t}_k = (t_k^1, \dots, t_k^{N_b})$, where $t_k^j \sim \mathcal{U}(0, 1)$ is the noise level assigned to block j . The corresponding corrupted sequence $\mathbf{x}^{t_k}$ is generated by masking each token x_i (where $j = \lceil i/B \rceil$) with probability t_k^j . The training objective averages the denoising

Algorithm 1 MMT Training

Require: Model p_{θ} , dataset $\mathcal{D}$, number of mask pairs $K/2$, block size B

```

1: for each training iteration do
2:   Sample minibatch  $\{\mathbf{x}\}$  from  $\mathcal{D}$ 
3:   // Generate  $K/2$  complementary mask pairs
4:   for  $v = 1$  to  $K/2$  do
5:     Sample noise  $\mathbf{t}_v \sim \mathcal{U}(0, 1)^{N/B}$ 
6:     Sample  $\mathbf{m}_v \in \{0, 1\}^N$ , with  $m_{v,i} \sim \text{Bernoulli}(t_v^{\lceil i/B \rceil})$ 
7:      $\mathbf{x}^{t_v} \leftarrow \mathbf{x} \odot (1 - \mathbf{m}_v) + \text{M} \odot \mathbf{m}_v$ 
8:      $\bar{\mathbf{x}}^{t_v} \leftarrow \mathbf{x} \odot \mathbf{m}_v + \text{M} \odot (1 - \mathbf{m}_v)$ 
9:   end for
10:  // Construct input with shared clean context
11:   $\mathbf{X} \leftarrow \text{Concat}(\mathbf{x}, \mathbf{x}^{t_1}, \bar{\mathbf{x}}^{t_1}, \dots, \mathbf{x}^{t_{K/2}}, \bar{\mathbf{x}}^{t_{K/2}})$ 
12:  Construct block-wise attention mask  $\mathbf{M}$  according to Eq. (6)
13:  // Forward and backward pass
14:  Compute denoising loss  $\mathcal{L}_{\text{MMT}}$  on  $\mathbf{X}$ 
15:  Update  $\theta$  via gradient descent on  $\mathcal{L}_{\text{MMT}}$ 
16: end for
```

losses over these K noised views:

$$\begin{aligned} \mathcal{L}_{\text{MMT}} &= \mathbb{E}_{\mathbf{x}, \{\mathbf{t}_k\}_{k=1}^K, \{\mathbf{x}^{t_k}\}_{k=1}^K} \left[\frac{1}{K} \sum_{k=1}^K \mathcal{J}_k(\mathbf{x}) \right], \\ \mathcal{J}_k(\mathbf{x}) &= -\sum_{i=1}^N \frac{\mathbb{I}[x_i^{t_k} = \text{M}]}{t_k^j} \ell_{\theta, i}^{\text{BDLM}}(t_k^j). \end{aligned} \quad (5)$$

where the expectation is taken over $\mathbf{x}$, the block-wise noise levels $\{\mathbf{t}_k\}_{k=1}^K$, and the corresponding corrupted sequences $\{\mathbf{x}^{t_k}\}_{k=1}^K$.

Context-sharing attention mechanism. To enable efficient parallel training of all K noised views while maintaining correct token visibility, we construct an attention mask $\mathbf{M} \in \{0, 1\}^{(K+1)N \times (K+1)N}$ over token indices. We use $k = 0$ for the clean sequence $\mathbf{x}$ and $k \in \{1, \dots, K\}$ to index the K corrupted sequences $\mathbf{x}^{t_k}$. Let (k, i) denote the i -th token of the k -th sequence. The attention mask entry $M_{(k', i')}^{(k, i)}$ determines whether token (k, i) can attend to token (k', i') :

$$M_{(k', i')}^{(k, i)} = \begin{cases} 1, & \text{if } k' = 0 \text{ and } \lceil i'/B \rceil < \lceil i/B \rceil, \\ 1, & \text{if } k' = k \text{ and } \lceil i'/B \rceil = \lceil i/B \rceil, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

This mask ensures that each masked context can attend to: (1) preceding blocks from the shared clean sequence, and (2) its own noisy block at the current position. The efficient training mask of MMT is illustrated in Figure 2.

	AR	BDLM	MMT (Ours)
Non-attn	$2PN$	$4PN$	$(1 + 1/K) \cdot 2PN$
Attn	$2N^2Ld$	$4N^2Ld$	$(1 + 1/K) \cdot 2N^2Ld$

Table 1: Average FLOPs per corrupted sequence. For MMT, the cost is amortized over K corrupted sequences, and $(1 + 1/K) \approx 1$ when K is reasonably large.

Complementary masking via context sharing.

We adopt the complementary masking strategy (Wu et al., 2025a). Instead of independently sampling K mask patterns, we sample $K/2$ pairs of complementary mask patterns. Crucially, unlike prior implementations that double the context computation cost, our context-sharing paradigm allows us to process complementary masks against a single clean context representation. This modification enables us to compute the paired loss jointly covering all tokens without normalization factors over individual noise levels. We provide the detailed formulation of the complementary masking objective in Appendix A.2.

Training algorithm. We summarize the complete training procedure of MMT in Algorithm 1. By concatenating the shared clean context with K complementary corrupted sequences and applying our specialized attention mask $\mathbf{M}$, we compute the denoising loss for all corrupted copies simultaneously in a single forward pass, achieving substantial efficiency gains.

3.3 Computational Complexity Analysis

We analyze the computational complexity of different language modeling paradigms. Let N denote the sequence length, P the non-embedding parameter count, L the number of transformer layers, and d the hidden dimension.

Per-token training cost. Following Chen et al. (2025), the FLOPs required to process one token with effective context length n , defined as the number of key/value tokens visible to the query token under the attention mask, is

$$C_{\text{token}}(n) = \underbrace{2P}_{\text{non-attn}} + \underbrace{4nLd}_{\text{attn}}. \quad (7)$$

Autoregressive Transformers. For causal language models generating a sequence of length N , the total FLOPs is

$$\begin{aligned} C_{\text{AR}} &= \sum_{t=1}^N C_{\text{token}}(t) = 2PN + \sum_{t=1}^N 4tLd \\ &= \underbrace{2PN}_{\text{non-attn}} + \underbrace{2N(N+1)Ld}_{\text{attn}}. \end{aligned} \quad (8)$$

Block Diffusion Language Models. Block diffusion models partition the sequence into $N_b = N/B$ blocks. At block j of the clean sequence $\mathbf{x}$, the model attends to all $(j-1)B$ preceding tokens plus the current block of size B . At block j of the noisy sequence $\mathbf{x}^t$, the model attends to all $(j-1)B$ preceding clean tokens plus the current noisy block of size B . That is, the j -th block of each sequence attends to jB tokens. The total FLOPs is

$$\begin{aligned} C_{\text{BDLM}} &= 2 \sum_{i=1}^N C_{\text{token}}(\lceil i/B \rceil B) \\ &= 2 \sum_{j=1}^{N_b} B \cdot C_{\text{token}}(jB) \\ &= 2[2PN_bB + 2B^2N_b(N_b+1)Ld] \\ &= 2 \times \left[\underbrace{2PN}_{\text{non-attn}} + \underbrace{2N(N+B)Ld}_{\text{attn}} \right]. \end{aligned} \quad (9)$$

MMT (Ours). Our context-sharing mechanism processes the clean context once and shares it across K corrupted contexts. We process $1 + K$ sequences in total, where the j -th block of each sequence attends to jB tokens. The total FLOPs is

$$\begin{aligned} C_{\text{MMT}} &= (1 + K) \sum_{j=1}^{N_b} B \cdot C_{\text{token}}(jB) \\ &= (1 + K) \times \left[\underbrace{2PN}_{\text{non-attn}} + \underbrace{2N(N+B)Ld}_{\text{attn}} \right]. \end{aligned} \quad (10)$$

The amortized cost per corrupted sequence is

$$\frac{C_{\text{MMT}}}{K} = (1+1/K) \times \left[\underbrace{2PN}_{\text{non-attn}} + \underbrace{2N(N+B)Ld}_{\text{attn}} \right]. \quad (11)$$

In typical training settings, the sequence length N is at least 2048 or 4096, while the block size B is usually set to small values such as 32 or 64 (Wu et al., 2025a). Since $B \ll N$, we have

B	K	ID (Val)	OOD Datasets							Mean OOD
		SlimPajama	PTB	WikiText	LM1B	AG News	PubMed	LAMBADA	ArXiv	
16	1	2.544 (baseline)	4.342	2.897	3.548	3.204	2.357	3.423	2.461	3.176 (baseline)
	2	2.515 ($\downarrow 0.029$)	4.275	2.865	3.501	3.152	2.330	3.383	2.439	3.135 ($\downarrow 0.041$)
	4	2.529 ($\downarrow 0.015$)	4.278	2.885	3.500	3.159	2.344	3.383	2.455	3.144 ($\downarrow 0.032$)
64	1	2.605 (baseline)	4.198	2.897	3.550	3.258	2.414	3.431	2.498	3.178 (baseline)
	2	2.569 ($\downarrow 0.036$)	4.149	2.855	3.534	3.231	2.377	3.414	2.474	3.148 ($\downarrow 0.030$)
	4	2.574 ($\downarrow 0.031$)	4.147	2.854	3.509	3.217	2.383	3.397	2.475	3.140 ($\downarrow 0.038$)

Table 2: Performance comparison of NELBO pretraining at the 1028M scale across different block sizes B and K values. SlimPajama serves as the In-Domain validation set, while the remaining 7 datasets evaluate Out-of-Domain (OOD) generalization. The lowest NELBO in each column for a given block size is highlighted in bold.

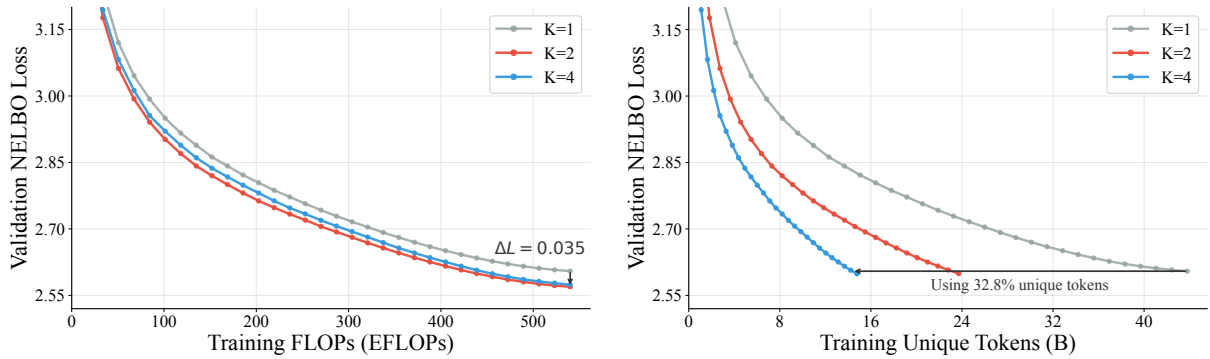


Figure 3: NELBO on SlimPajama validation set throughout pretraining of the 1028M model with block size $B = 64$, measured by training FLOPs (left) and consumed unique tokens (right). MMT reaches lower NELBO under both metrics with multiple mask patterns, demonstrating improved compute and token efficiency from context sharing.

$N(N + B) \approx N^2$ and $N(N + 1) \approx N^2$. Table 1 summarizes the average training FLOPs per corrupted sequence under this approximation. As shown, MMT significantly improves FLOPs efficiency compared to block diffusion, which is a special case of our method with $K = 1$.

4 Experiment

We conduct experiments in two training scenarios: pretraining block diffusion models from scratch and AR-to-Diffusion adaptation from a pretrained autoregressive model.

4.1 Block Diffusion Model Pretraining

For pretraining, we build upon the codebase of SMDM (Nie et al., 2025a) and adapt it to the block diffusion setting. The vanilla single-mask ($K=1$) configuration serves as our baseline. **All pretraining comparisons use the same training-time budget for each choice of K and block size B .**

Models. We train masked diffusion language models from scratch at scales up to 1028M parameters. All models are trained on the SlimPajama (Soboleva et al., 2023) dataset. We use a

context length of 2048 tokens and fix the global batch size to 256 *masked copies* per optimizer step; for $K > 1$, this corresponds to fewer unique sequences because each sequence contributes multiple masked views. We evaluate two block sizes, $B = 16$ and $B = 64$, to cover both fine- and coarse-grained diffusion blocks. Additional optimization details are provided in Appendix B.1.

Benchmarks. We evaluate the pretrained models by NELBO on both in-domain (ID) and out-of-domain (OOD) corpora. For the ID evaluation, we report NELBO on the held-out validation split of SlimPajama, which directly reflects how well the model fits the pretraining distribution. In parallel, following BD3-LMs (Arriola et al., 2025), we assess the zero-shot likelihood generalization of the same models on seven OOD datasets spanning a broad spectrum of textual genres beyond the training distribution. The detailed corpus list is provided in Appendix B.1. Since the NELBO is an expectation over block-wise noise levels and mask patterns, all NELBO numbers reported in this paper are estimated with 10 Monte Carlo samples per sequence.

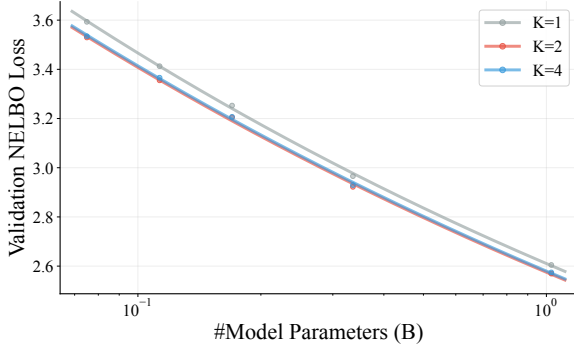


Figure 4: SlimPajama validation loss across five model scales (75M, 113M, 170M, 336M, and 1028M parameters) for $K \in \{1, 2, 4\}$ at $B = 64$. The improvement of MMT ($K > 1$) over the baseline persists consistently across all model scales.

Results. As shown in Table 2, across both block sizes, every $K > 1$ configuration of MMT improves over the baseline on the in-domain validation set as well as on the average OOD NELBO, confirming that context sharing brings consistent gains. On OOD generalization, the best mean NELBO is attained by $K = 2$ at $B = 16$ and by $K = 4$ at $B = 64$. We attribute this trend to the fact that a larger block size leaves more positions per sequence to be corrupted, providing more room for diverse mask patterns and thus benefiting more from a larger K .

To further examine the training dynamics, we uniformly sample 32 checkpoints along the training trajectory and evaluate their NELBO on the SlimPajama validation set. Figure 3 plots the $B = 64$ trajectories against training FLOPs and unique tokens consumed (see Appendix B.2 for $B = 16$). The FLOPs comparison directly reflects real training cost, since equal FLOPs translate to equal wall-clock training time. Under this matched-compute view, MMT attains a lower NELBO than the $K = 1$ baseline across all mask patterns, confirming that the efficiency gain from context sharing substantially outweighs the reduction in unique data exposure. The unique-token view further shows that a single mask sample per sequence under-utilizes the supervisory signal in each sequence, and that drawing additional mask patterns yields substantially greater gains from the same unique data.

Scaling Across Model Sizes. As shown in Figure 4, across model sizes from 75M to 1028M, $K = 2/4$ consistently outperform the baseline, and the gap does not shrink as the model grows. This indicates that the benefit of context sharing is

not tied to a particular scale but is a robust property of the multi-mask training paradigm.

Complementary Masking Ablation. We ablate complementary masking at the 336M scale by comparing independently sampled masks with complementary mask pairs for the same K . As shown in Table 3, complementary masking yields lower validation NELBO. This motivates our use of even K values in the main experiments, so that all mask patterns can be organized into complementary pairs.

Metric	$K = 2$		$K = 4$	
	w/o CM	w/ CM	w/o CM	w/ CM
SlimPajama	2.935	2.921	2.937	2.928
Mean OOD	3.510	3.508	3.509	3.498

Table 3: Ablation of complementary masking (CM) at the 336M scale with block size $B = 64$. Detailed per-dataset OOD results are provided in Appendix B.4.

Training Objective Ablation. We compare our default objective against the original block diffusion NELBO at the 336M scale with $B=64$ and $K=1$ in Table 4. **Original Block Diffusion NELBO** is the SMDM-style per-token NELBO estimator (Nie et al., 2025a), which importance-weights each masked token by $1/p_{\text{mask}}$. Although this estimator is unbiased, the weight becomes very large when t is small, leading to high-variance gradients and less stable optimization. Our actual training objective is **Equal-weight CE**, the same objective as Fast-dLLM v2 (Wu et al., 2025a): it removes the $1/p_{\text{mask}}$ importance weight and optimizes an equal-weight token-level cross-entropy over masked tokens. As shown in Table 4, this surrogate objective is more stable and achieves better NELBO, so we use Equal-weight CE as the default training objective in all main experiments. Additional variants that stabilize or modify the original block diffusion NELBO are discussed in Appendix A.3.

Method	ID (Val)	Mean OOD
	SlimPajama	
Equal-weight CE	2.964 (+0.000)	3.533 (+0.000)
Original Block Diffusion NELBO	3.032 (+0.068)	3.601 (+0.068)

Table 4: Comparison between our training objective and the original block diffusion NELBO at the 336M scale ($B=64$, $K=1$). We report NELBO on the in-domain validation set and average across 7 OOD datasets. Detailed OOD results are provided in Appendix A.3.

Method	GSM8K	MATH	MMLU	GPQA	IFEval	MBPP	HumanEval	Avg.	Time (h)
Fast-dLLM v2	62.0	38.1	55.1	27.7	47.0	50.0	43.9	46.3	-
MMT, $K=1$	69.96 $\pm$ 0.34	41.43 $\pm$ 0.36	51.80 $\pm$ 0.14	26.00 $\pm$ 0.81	61.17 $\pm$ 1.25	48.82 $\pm$ 1.22	41.08 $\pm$ 2.44	48.61 $\pm$ 0.13	29.67
MMT, $K=2$	70.00 $\pm$ 0.58	41.28 $\pm$ 0.42	51.79 $\pm$ 0.86	26.15 $\pm$ 2.59	60.33 $\pm$ 0.79	49.09 $\pm$ 1.97	43.01 $\pm$ 0.99	48.81 $\pm$ 0.16	22.42
MMT, $K=4$	69.44 $\pm$ 0.81	40.73 $\pm$ 0.14	51.19 $\pm$ 0.49	26.95 $\pm$ 1.07	59.81 $\pm$ 0.50	46.33 $\pm$ 1.23	42.15 $\pm$ 0.92	48.09 $\pm$ 0.14	18.80

Table 5: Performance of MMT under different numbers of mask patterns K in the 1.5B AR-to-Diffusion setting. All MMT variants use a matched-step protocol of 6,000 optimizer steps. Mean and standard deviation are computed across three seeds, after averaging the final 10 checkpoints of each run. Per-seed results are in Appendix D.2. Fast-dLLM v2 results are cited from Wu et al. (2025a), retaining their reported single-decimal precision.

4.2 Speedup Analysis

Setup. We benchmark our largest pretraining setup, a 1028M-parameter model with $B = 64$, on 8 A800-80GB GPUs with FSDP ZeRO-3 and bfloat16. Each optimizer step processes a global batch of 256 *masked copies*, with the per-device batch of 32 split via gradient accumulation into $32/\mu$ micro-batches of size μ (also in masked copies). We report $\mu = 8$, the largest value that fits the $K = 1$ baseline within 80 GB, and additionally $\mu = 16$ for $K \in \{4, 8\}$, where context sharing saves enough activation memory. Step time is averaged over steps 100–2500 to capture steady-state throughput, excluding FSDP initialization and external logger startup overhead.

Training Speedup. As reported in Table 6, at fixed $\mu = 8$, increasing K simultaneously reduces step time and memory, yielding speedups from $1.30\times$ at $K = 2$ to $1.72\times$ at $K = 8$, consistent with the theoretical limit $2/(1 + 1/K)$. The saved memory can optionally be reinvested in a larger micro-batch: $\mu = 16$ reaches up to $1.75\times$ speedup. Appendix C.2 provides the complete memory and feasibility analysis.

K	μ	Step time (s)	Speedup	Mem (GiB)
1	8	5.23	1.00 $\times$	66.0
2	8	4.02	1.30 $\times$	53.7
4	8	3.40	1.54 $\times$	46.4
8	8	3.04	1.72 $\times$	43.3
4	16	3.26	1.60 $\times$	79.2
8	16	2.99	1.75 $\times$	73.4

Table 6: Training efficiency of MMT at the 1028M scale. The micro-batch size μ is measured in masked copies. Full results at longer contexts are in Appendix C.2.

4.3 AR-to-Diffusion Adaptation

We further validate MMT in the AR-to-Diffusion setting, where a pretrained autoregressive model is

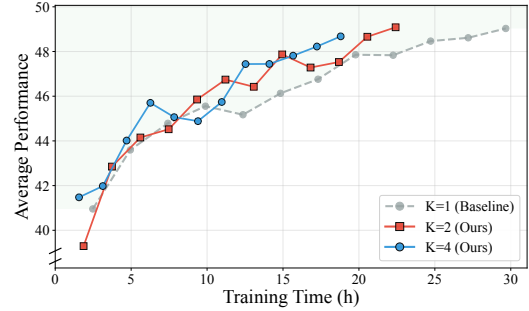


Figure 5: Training efficiency: Time vs. Performance. Our method establishes a superior Pareto frontier by achieving comparable performance with less training time.

adapted into a block diffusion language model.

Models. We initialize our model from Qwen2.5-1.5B-Instruct (Yang et al., 2024) and fine-tune it into a block diffusion language model following the training recipe of Fast-dLLM v2 (Wu et al., 2025a). The fine-tuning data is sampled from the LLaMA-Nemotron post-training dataset (Bercovich et al., 2025) and the IFEval dataset (Zhou et al., 2023). All models are trained for 6000 optimizer steps. Detailed data mixture, token budgets, and optimization hyperparameters are provided in Appendix D.1.

Benchmarks. We evaluate general ability on MMLU (Hendrycks et al., 2021a) and GPQA (Rein et al., 2024), mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), and code generation on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). All benchmarks are evaluated in the zero-shot setting, except MMLU which uses the 5-shot setting.

Results. Table 5 reports the AR-to-Diffusion results. $K=2$ achieves comparable downstream quality to the $K=1$ baseline while using only about 75.6% of its wall-clock training time. Pushing fur-

ther to $K=4$ reduces the training time to roughly 63.4% of the baseline, with a modest decrease in the average score. As visualized in Figure 5, MMT thus establishes a strictly better performance-efficiency Pareto frontier.

5 Conclusion

We presented MMT, a simple training paradigm for Block Diffusion Language Models that mitigates their low computational utilization. Specifically, MMT processes K masked views of the same sequence in a single forward pass through context-sharing sparse attention, lifting utilization from 50% to $K/(1+K)$. Across both pretraining and AR-to-Diffusion adaptation, MMT consistently improves training efficiency for Block Diffusion Language Models. In pretraining, under matched compute, the efficiency gain from context-shared multi-mask training decisively outweighs the cost of reduced unique-data exposure. Our AR-to-Diffusion adaptation experiments instead compare models under matched optimizer steps and report the corresponding wall-clock training time. Together, these findings establish mask diversity as a more compute-efficient axis for scaling Block Diffusion Language Models.

Limitations

While MMT delivers substantial efficiency gains for Block Diffusion Language Models, our study has several limitations that we leave for future investigation.

Integration with Reinforcement Learning for Language Models. In this work, we focus on the pretraining and supervised fine-tuning stages, and do not extend MMT to reinforcement learning (RL). Nevertheless, our context-sharing attention mask is naturally suited to accelerating diffusion-based RL: estimating the likelihood of a generated response requires Monte-Carlo sampling over multiple mask patterns, which is computationally expensive under the standard training pattern. By packing these mask patterns into a single forward pass with our sparse attention mechanism, the cost of likelihood estimation during policy optimization could in principle be substantially reduced. We have not empirically validated this hypothesis, and leave a thorough study of MMT for diffusion-based language RL to future work.

Extension to Continuous Modalities. Our experiments are restricted to discrete language modeling, and we do not evaluate MMT on continuous modalities. A growing line of work, including continuous diffusion language models and block-wise autoregressive visual generators that synthesize image or video patches in sequential blocks, models data in a continuous space and has recently attracted considerable attention. Conceptually, the core idea of MMT, which amortizes the cost of a shared clean context across multiple masked views, does not depend on whether the modeled tokens or patches are discrete or continuous, and could in principle accelerate the training of such models as well. We have not empirically verified its effectiveness or scalability in continuous-valued settings, and view this as another promising direction for future research.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2020YFA0804503, the National Natural Science Foundation of China under Grant No. 62272264 and Grant No. 62576186, and the Tsinghua University Initiative Scientific Research Program.

References

- Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shafaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. [Llama-nemotron: Efficient reasoning models](#). *Preprint*, arXiv:2505.00949.
- 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, and 12 others. 2025. [Llada 2.0: Scaling up diffusion language models to 100b](#). *Preprint*, arXiv:2512.15745.
- Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, and 95 others. 2021. [On the opportunities and risks of foundation models](#). *Preprint*.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Proceedings of Interspeech 2014*, pages 2635–2639.
- 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, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. 2023. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*.
- Yingfa Chen, Yutong Wu, Chenyang Song, Zhen Leng Thai, Xingyu Shen, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. [Cost-optimal grouped-query attention for long-context modeling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5360–5376.
- Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. 2026. [SDAR: A synergistic diffusion-AutoRegression paradigm for scalable sequence generation](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 22058–22075.
- 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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621.
- Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. 2024. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*.
- Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*.
- Google DeepMind. 2025. Gemini diffusion. <https://blog.google/technology/google-deepmind/gemini-diffusion/>. Accessed: 2026-05-11.
- Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2023. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11575–11596.
- Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, and 5 others. 2021. Pre-trained models: Past, present and future. *AI Open*, 2:225–250.
- Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. 2023. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4521–4534.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030.
- Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. 2022. Autoregressive diffusion models. In *International Conference on Learning Representations*.
- Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465.
- Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. 2025. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*.
- Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*.
- Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. 2023. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.
- Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. 2025a. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*.
- Jinjie Ni, Qian Liu, Chao Du, Longxu Dou, Hang Yan, Zili Wang, Tianyu Pang, and Michael Qizhe Shieh. 2025b. Training optimal large diffusion language models. *arXiv preprint arXiv:2510.03280*.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up masked diffusion models on text. In *The Thirteenth International Conference on Learning Representations*.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- OpenAI. 2023. [GPT-4 technical report](#). *Preprint*.
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534.
- William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.
- Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Kateřina Fragkiadaki, and Deepak Pathak. 2025. Diffusion beats autoregressive in data-constrained settings. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. Slimpajama: A 627b token, cleaned and deduplicated version of redpajama. *Cerebras blog post*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr.

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

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, and 3 others. 2025. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*.

Robin Strudel, Corentin Tallec, Florent Alché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibao Fang, Tianyu Guo, Kai Han, Chao Xu, Hanting Chen, Xinghao Chen, and Yunhe Wang. 2025. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*.

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

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Training Objectives and Theoretical Analysis

A.1 Unbiasedness of the Idealized MMT Estimator

We prove that the *idealized* MMT objective of Eq. (5), in which the K mask patterns are sampled independently and every noised view is scored with the original importance-weighted objective, is an unbiased estimator of the BDLM objective. Recall that the BDLM loss for a single sample $\mathbf{x}$ is:

$$\begin{aligned} \mathcal{L}_{\text{BDLM}}(\mathbf{x}) &= -\mathbb{E}_{\mathbf{t}, \mathbf{x}^{\mathbf{t}}} \left[\sum_{i=1}^N \frac{\mathbb{I}[x_i^{\mathbf{t}} = \mathbf{M}]}{t_j} \ell_{\theta, i}^{\text{BDLM}}(t_j) \right], \quad (12) \\ \ell_{\theta, i}^{\text{BDLM}}(t_j) &= \log p_{\theta} \left(x_i \mid \mathbf{b}_{< j}, \mathbf{b}_j^{t_j} \right). \end{aligned}$$

where $j = \lceil i/B \rceil$ is the block index for token i , and the expectation is over the block-wise noise levels $\mathbf{t} = (t_1, \dots, t_{N_b})$ with $t_j \sim \mathcal{U}(0, 1)$ and the stochastic masking process.

Theorem A.1 (Unbiasedness of the idealized estimator). *Assume that (i) the K block-wise noise levels $\{\mathbf{t}_k\}_{k=1}^K$ are i.i.d. samples from $\mathcal{U}(0, 1)^{N_b}$ and the corresponding corrupted sequences $\{\mathbf{x}^{\mathbf{t}_k}\}_{k=1}^K$ are obtained by independent Bernoulli masking; and (ii) every noised view is scored with the importance-weighted objective of Eq. (4), i.e., each masked token is weighted by the inverse of the noise level of its own block. Then:*

$$\begin{aligned} -\mathbb{E}_{\{\mathbf{t}_k\}, \{\mathbf{x}^{\mathbf{t}_k}\}} \left[\frac{1}{K} \sum_{k=1}^K \sum_{i=1}^N \frac{\mathbb{I}[x_i^{\mathbf{t}_k} = \mathbf{M}]}{t_k^j} \ell_{\theta, i}^{\text{BDLM}}(t_k^j) \right] &= \mathcal{L}_{\text{BDLM}}(\mathbf{x}). \quad (13) \end{aligned}$$

Proof. Define the loss per noised view:

$$\ell_k(\mathbf{x}) \triangleq - \sum_{i=1}^N \frac{\mathbb{I}[x_i^{\mathbf{t}_k} = \mathbf{M}]}{t_k^j} \ell_{\theta, i}^{\text{BDLM}}(t_k^j). \quad (14)$$

By linearity of expectation:

$$\mathbb{E} \left[\frac{1}{K} \sum_{k=1}^K \ell_k(\mathbf{x}) \right] = \frac{1}{K} \sum_{k=1}^K \mathbb{E}[\ell_k(\mathbf{x})]. \quad (15)$$

Since each $(\mathbf{t}_k, \mathbf{x}^{\mathbf{t}_k})$ is sampled from the same distribution as $(\mathbf{t}, \mathbf{x}^{\mathbf{t}})$ in BDLM:

$$\begin{aligned} \mathbb{E}[\ell_k(\mathbf{x})] &= \mathbb{E}_{\mathbf{t}, \mathbf{x}^{\mathbf{t}}}[\ell(\mathbf{x})] \\ &= \mathcal{L}_{\text{BDLM}}(\mathbf{x}), \quad \forall k \in \{1, \dots, K\}. \end{aligned} \quad (16)$$

Therefore:

$$\begin{aligned} \mathbb{E} \left[\frac{1}{K} \sum_{k=1}^K \ell_k(\mathbf{x}) \right] &= \frac{1}{K} \cdot K \cdot \mathcal{L}_{\text{BDLM}}(\mathbf{x}) \\ &= \mathcal{L}_{\text{BDLM}}(\mathbf{x}). \end{aligned} \quad (17)$$

□

Remark A.2 (Variance and compute). Under the assumptions of Theorem A.1, MMT introduces no bias—it simply uses K Monte Carlo samples to estimate the same expected loss as BDLM, with each mask pattern sampled independently while sharing the same clean context $\mathbf{x}$. Crucially, while a naive K -sample Monte Carlo estimate would require K independent BDLM forward passes (with a total cost of $2K$ context-length sequences), MMT obtains the same estimate in a single forward pass over only $1+K$ sequences via context sharing (see Sec. 3.3), thereby reducing mask-sampling variance at *substantially lower* compute than naive K -sample BDLM.

Remark A.3 (Scope of the unbiasedness result). Theorem A.1 applies to the idealized objective of Eq. (5). The main experiments instead use complementary masking (Appendix A.2) and equal-weight cross-entropy (Appendix A.3), which do not satisfy assumptions (i) and (ii). This practical recipe is biased relative to the BDLM objective; we adopt it for lower gradient variance and better empirical NELBO.

A.2 Complementary Masking Formulation

As introduced in Section 3.2, we adopt the complementary masking strategy (Wu et al., 2025a) within our context-sharing paradigm. Instead of independently sampling K mask patterns, we sample $K/2$ pairs of complementary mask patterns. The training objective of MMT becomes:

$$\mathcal{L}_{\text{MMT}} = -\mathbb{E}_{\{\mathbf{t}_v\}_{v=1}^{K/2}, \{\mathbf{x}^{\mathbf{t}_v}\}_{v=1}^{K/2}} \left[\frac{1}{K} \sum_{v=1}^{K/2} \mathcal{L}_{\text{pair}}^{(v)} \right]. \quad (18)$$

Each pair of complementary masks consists of an original mask pattern m and its inverse $\bar{m} = 1 - m$, generating two noisy sequences $\mathbf{x}^{\mathbf{t}_v}$ and $\bar{\mathbf{x}}^{\mathbf{t}_v}$, where

$$\bar{x}_i^{\mathbf{t}_v} = \begin{cases} x_i & \text{if } x_i^{\mathbf{t}_v} = \text{M}, \\ \text{M} & \text{otherwise.} \end{cases} \quad (19)$$

The paired loss is then defined as

$$\begin{aligned} \mathcal{L}_{\text{pair}}^{(v)} &= \sum_{i=1}^N [x_i^{\mathbf{t}_v} = \text{M}] \log p_\theta(x_i \mid \mathbf{b}_{<j}, \mathbf{b}_j^{\mathbf{t}_v}) \\ &\quad + \sum_{i=1}^N [\bar{x}_i^{\mathbf{t}_v} = \text{M}] \log p_\theta(x_i \mid \mathbf{b}_{<j}, \bar{\mathbf{b}}_j^{\mathbf{t}_v}), \end{aligned} \quad (20)$$

where $\bar{\mathbf{b}}_j^{\mathbf{t}_v}$ is the j -th noisy block of $\bar{\mathbf{x}}^{\mathbf{t}_v}$ (with $j = \lceil i/B \rceil$). We eliminate the normalization factor $1/t_v^j$ following Wu et al. (2025a), as the complementary masks jointly cover all N tokens.

A.3 Detailed Training Objective Ablation

Table 7 provides the full training objective ablation at the 336M scale with $B=64$ and $K=1$. In addition to the comparison between our default Equal-weight CE objective and the Original Block Diffusion NELBO in Table 4, we evaluate two related variants. We organize all four objectives below as successive modifications of the **Original Block Diffusion NELBO**, the SMDM-style per-token NELBO estimator (Nie et al., 2025a) that importance-weights each masked token by $1/p_{\text{mask}}$.

Stabilized NELBO. This objective formalizes a variance-reduction modification on top of the Original Block Diffusion NELBO. Let $M_i := \mathbb{I}[x_i^{\mathbf{t}} = \text{M}]$ denote the token-level mask indicator, and let

$$m_j := \sum_{i \in \text{block}_j} M_i, \quad (21)$$

denote the empirical number of masked tokens in block j , with $\mathbb{E}_{M|t_j}[m_j] = B \cdot t_j$ under i.i.d. Bernoulli masking. We further abbreviate the BDLM log-likelihood in Eq. (4) as $\ell_i := \ell_{\theta,i}^{\text{BDLM}}(t_j)$ for token i in block j . In our implementation, the Original Block Diffusion NELBO is normalized by the total noisy-token count N per sample, following the per-token form used in SMDM (Nie et al., 2025a). Using $N = N_b \cdot B$, this

objective admits a block-wise regrouping:

$$\begin{aligned}\mathcal{L}_{\text{ori}} &= -\mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \frac{M_i \ell_i}{t_j} \right] \\ &= -\mathbb{E} \left[\frac{1}{N_b} \sum_{j=1}^{N_b} \frac{1}{B t_j} \sum_{i \in \text{block}_j} M_i \ell_i \right],\end{aligned}\quad (22)$$

i.e., the block-internal weight $1/(B t_j)$ is the inverse *expected* mask count $1/\mathbb{E}[m_j | t_j]$. Stabilized NELBO substitutes this expected count $B t_j$ with the empirically sampled count m_j :

$$\mathcal{L}_{\text{stab}} = -\mathbb{E} \left[\frac{1}{N_b} \sum_{j=1}^{N_b} \frac{\mathbb{I}[m_j > 0]}{m_j} \sum_{i \in \text{block}_j} M_i \ell_i \right], \quad (23)$$

with the convention that blocks containing no masked tokens ($m_j = 0$) contribute zero. Since $\mathbb{E}_{M|t_j}[m_j] = B t_j$, the substitution preserves the first moment. It also keeps the variance bounded because $1/m_j \leq 1$ whenever $m_j \geq 1$, whereas $1/(B t_j)$ diverges as $t_j \rightarrow 0$. Under i.i.d. Bernoulli masking, however, this estimator is biased in the small- t regime, since blocks with $m_j = 0$ contribute zero yet still occupy a slot in the constant block-count denominator N_b , leading to a small- t underestimation of the true NELBO.

Shared- t CE. This variant starts from Stabilized NELBO and changes how t is sampled: instead of drawing an independent t_j for each block, we follow SDAR (Cheng et al., 2026) and share a single noise level t across all blocks within a sequence. Under this per-sample t , all blocks in a sequence share the same p_{mask} , so the block-level heterogeneity that the $1/m_j$ normalization in Eq. (23) was designed to absorb disappears, and the per-block normalization naturally collapses to a sequence-level one. The loss objective consequently reduces to an equal-weight cross-entropy over masked tokens, normalized by the total masked-token count $\sum_{j=1}^{N_b} m_j$ in each sequence.

Equal-weight CE. This variant restores the per-block t_j schedule of Stabilized NELBO while keeping the sequence-level CE form that Shared- t CE arrived at: each masked token contributes equally to the cross-entropy, and the loss is normalized by the sample-level mask count $\sum_{j=1}^{N_b} m_j$ rather than the block-level m_j . This is the same objective as Fast-dLLM v2 (Wu et al., 2025a) and serves as our default in the main experiments.

As shown in Table 7, both Stabilized NELBO and Equal-weight CE substantially improve over Original Block Diffusion NELBO, confirming that replacing the high-variance $1/p_{\text{mask}}$ weight with empirical mask counts is beneficial. Comparing Equal-weight CE against Shared- t CE, which differ only in whether t is sampled per block or per sample, the per-block schedule yields a clear additional improvement, suggesting that the diversity of block-level noise levels provides useful training signal. Equal-weight CE combines both improvements and attains the best NELBO on both the in-domain validation set and the OOD average.

B Pretraining Details and Additional Results

B.1 Experimental Details

Model configurations. All models adopt the LLaMA-style Transformer architecture of SMDM (Nie et al., 2025a), with rotary position embeddings, RMSNorm, SwiGLU feed-forward layers, and a 32,000-entry vocabulary. Following Nie et al. (2025a), model scales are named by non-embedding parameter count (Table 8).

Optimization. For all pretraining runs, we use the AdamW optimizer (Loshchilov and Hutter, 2019) with a peak learning rate of $2e - 4$. The learning rate is warmed up for the first 1% of the total training steps and then decayed with a cosine schedule.

Evaluation Corpora. For in-domain evaluation, we report NELBO on the held-out validation split of SlimPajama (Soboleva et al., 2023). For out-of-domain evaluation, following BD3-LMs (Arriola et al., 2025), we evaluate zero-shot likelihood generalization on Penn Treebank (PTB) (Marcus et al., 1993), WikiText-103 (Merity et al., 2017), the One Billion Word Benchmark (LM1B) (Chelba et al., 2014), LAMBADA (Paperno et al., 2016), AG News (Zhang et al., 2015), and the PubMed and ArXiv subsets of the Scientific Papers corpus (Cohan et al., 2018).

B.2 Additional Pretraining Curves

Here, we present the pretraining curves of the 1028M model for block size $B = 16$. Figure 6 plots the Negative Evidence Lower Bound (NELBO) on the SlimPajama validation set against both training FLOPs and the number of consumed unique tokens. Consistent with the observations

Method	ID (Val)	Out-of-Domain (OOD)							Mean OOD
	SlimPajama	PTB	WikiText	LM1B	AG News	PubMed	LAMBADA	ArXiv	
Original Block Diffusion NELBO	3.032 (baseline)	4.549	3.326	3.906	3.857	2.892	3.758	2.917	3.601 (baseline)
Stabilized NELBO	2.996 ($\downarrow 0.036$)	4.487	3.301	3.868	3.810	2.856	3.736	2.883	3.563 ($\downarrow 0.038$)
Shared- t CE	3.034 ($\uparrow 0.002$)	4.560	3.337	3.903	3.848	2.899	3.748	2.916	3.602 ($\uparrow 0.001$)
Equal-weight CE	2.964 ($\downarrow 0.068$)	4.465	3.264	3.857	3.774	2.809	3.723	2.837	3.533 ($\downarrow 0.068$)

Table 7: Full training objective ablation at the 336M scale ($B=64$, $K=1$). We report NELBO on the in-domain validation set (SlimPajama), per-dataset NELBO on 7 OOD datasets, and the average across them.

Scale	L	d	d_{ff}	Heads	Non-emb.	Total
75M	16	640	1600	10	75.4M	116.3M
113M	12	768	3072	12	113.3M	162.4M
170M	18	768	3072	12	169.9M	219.1M
336M	20	1024	4096	16	335.6M	401.1M
1028M	20	1792	7168	14	1027.7M	1142.4M

Table 8: Architectural hyperparameters of the five pre-training scales. L is the number of Transformer layers, d the hidden dimension, and d_{ff} the feed-forward dimension.

Model Size	FLOPs (10^{18})	$K=1$	$K=2$	$K=4$
75M	3.03	3.593	3.530	3.535
113M	6.84	3.413	3.355	3.366
170M	15	3.253	3.204	3.207
336M	60	2.964	2.921	2.928
1028M	540	2.605	2.569	2.574

Table 9: Training FLOPs budgets and raw SlimPajama validation losses across five model scales for $K \in \{1, 2, 4\}$ at block size $B=64$.

at $B=64$, MMT with multiple mask patterns ($K > 1$) consistently achieves a lower NELBO than the single-mask baseline ($K=1$) under both matched-compute and matched-data comparisons.

B.3 Parameter Scaling Raw Results

Table 9 reports the SlimPajama validation losses used in the parameter scaling study shown in Figure 4. We allocate training FLOPs across model sizes following Chinchilla (Hoffmann et al., 2022) ($C \propto N^2$).

B.4 Detailed Complementary Masking Results

Table 10 reports the per-dataset NELBO underlying the complementary masking ablation summarized in Table 3. For both $K=2$ and $K=4$, complementary masking (w/ CM) improves over independently sampled masks (w/o CM) on the SlimPajama validation set and on the mean OOD NELBO, and the gain is consistent across the majority of OOD corpora.

C Efficiency and Practical Selection of K

C.1 Trade-off and Practical Selection of K

Increasing K amortizes the shared clean-context computation over more corrupted views, but under a fixed compute budget it also reduces the number of unique clean sequences seen during training. Let C_0 be the cost of processing one sequence copy. An MMT bundle costs $(K+1)C_0$ and provides K denoising views. Relative to the $K=1$ baseline, the denoising-view throughput and unique-data exposure under the same theoretical FLOPs budget are

$$R_{\text{view}}(K) = \frac{2K}{K+1}, \quad R_{\text{unique}}(K) = \frac{2}{K+1}. \quad (24)$$

These quantities evolve differently as K increases. Moving from $K=1$ to $K=2$ raises denoising-view throughput to $1.333\times$, while retaining 66.7% of the baseline unique-token exposure. In contrast, moving from $K=4$ to $K=8$ raises view throughput only from $1.600\times$ to $1.778\times$, while reducing unique-token exposure from 40.0% to 22.2%.

The measured wall-clock results exhibit the same diminishing-return pattern. At fixed $\mu=8$, the measured speedups $S(K)$ over $K=1$ are $1.30\times$, $1.54\times$, and $1.72\times$ for $K=2, 4$, and 8 , respectively (Table 12). Because each step sees only $1/K$ as many unique sequences, the corresponding relative unique-token throughput under a fixed wall-clock budget is $S(K)/K$, namely $0.650\times$, $0.385\times$, and $0.215\times$. Thus, larger K continues to improve systems throughput, but each increment brings a smaller speed gain while sacrificing a substantial fraction of unique training data.

Table 11 confirms the consequence for model quality. $K=2$ achieves the best in-domain NELBO and $K=4$ the best mean OOD NELBO, while increasing K to 6 or 8 weakens the benefit of MMT. These results support $K=2$ as a default, with $K=4$ preferable when OOD NELBO is the primary target.

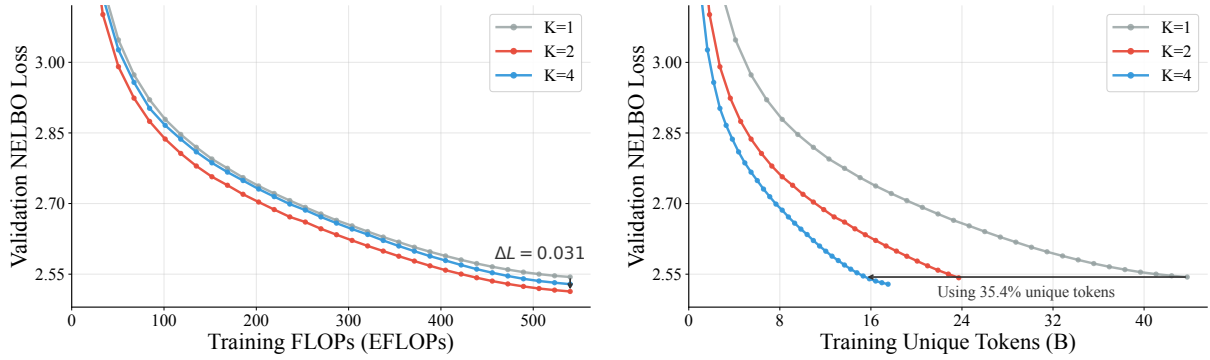


Figure 6: NELBO on SlimPajama validation set throughout pretraining of the 1028M model with block size $B = 16$, measured by training FLOPs (left) and consumed unique tokens (right). MMT reaches lower NELBO under both metrics with multiple mask patterns, demonstrating improved compute and token efficiency from context sharing.

Method	ID (Val)	Out-of-Domain (OOD)							Mean OOD
	SlimPajama	PTB	WikiText	LM1B	AG News	PubMed	LAMBADA	ArXiv	
$K = 2$, w/o CM	2.935	4.456	3.246	3.836	3.725	2.775	3.716	2.814	3.510
$K = 2$, w/ CM	2.921	4.515	3.232	3.837	3.713	2.759	3.698	2.799	3.508
$K = 4$, w/o CM	2.937	4.470	3.239	3.831	3.735	2.778	3.691	2.816	3.509
$K = 4$, w/ CM	2.928	4.450	3.235	3.811	3.715	2.763	3.698	2.813	3.498

Table 10: Full complementary masking (CM) ablation at the 336M scale with block size $B = 64$. We report NELBO on the in-domain validation set (SlimPajama), per-dataset NELBO on 7 OOD datasets, and the average across them.

C.2 Detailed Memory Scalability

We measure the micro-batch size μ in *masked copies*. At fixed μ , the $K = 1$ baseline processes approximately $2\mu N$ token representations per micro-batch, whereas MMT shares each clean sequence among K corrupted copies and processes approximately $\mu(1 + 1/K)N$. Consequently, increasing K reduces activation memory at the same μ ; the saved memory can instead be used to increase μ and reduce gradient-accumulation overhead.

Table 12 separates the effect of K from that of μ . At the controlled $\mu = 8$, memory decreases monotonically from 66.0 GiB at $K = 1$ to 43.3 GiB at $K = 8$. Doubling to $\mu = 16$ is infeasible for $K \in \{1, 2\}$ on the same hardware but becomes feasible once sufficient memory is saved at $K \in \{4, 8\}$.

The same pattern persists at longer contexts (Table 13). At 4k, doubling μ from 4 to 8 exceeds the A800 memory limit for $K \in \{1, 2\}$ but fits at $K = 4$. At 8k on B200 GPUs, $\mu = 8$ is infeasible for $K = 1$ but fits for $K \in \{2, 4\}$. Across all tested context lengths, increasing K consistently lowers memory at fixed μ and can make larger micro-batches feasible.

D AR-to-Diffusion Details and Results

D.1 Training Setup and Data Mixture

Training setup. For AR-to-Diffusion adaptation, we use block size $B = 32$, context length 2048, and a global batch size of 512 masked copies. All runs are trained for 6000 optimizer steps with AdamW (Loshchilov and Hutter, 2019). The learning rate is warmed up for the first 500 steps and then decayed with a cosine schedule, with a peak learning rate of $4e - 5$. Under this fixed masked-copy budget, the $K=1$ run consumes approximately 6.29B unique tokens, while $K=2$ and $K=4$ consume approximately 3.15B and 1.57B unique tokens, respectively.

Data mixture. We sample fine-tuning examples from the SFT portion of the LLaMA-Nemotron post-training dataset (Bercovich et al., 2025). Table 14 reports the normalized sampling mixture used in our AR-to-Diffusion runs.

D.2 Multi-seed Evaluation

We train the Qwen2.5-1.5B-Instruct variants with three independent random seeds for each K . For each seed, we first average the evaluations over the final 10 checkpoints and then compute the mean

K	R_{view}	R_{unique}	ID (Val)	Out-of-Domain (OOD)							Mean OOD
			SlimPajama	PTB	WikiText	LM1B	AG News	PubMed	LAMBADA	ArXiv	
1	1.000×	1.000×	2.964	4.465	3.264	3.857	3.774	2.809	3.723	2.837	3.533
2	1.333×	0.667×	2.921	4.515	3.232	3.837	3.713	2.759	3.698	2.799	3.508
4	1.600×	0.400×	2.928	4.450	3.235	3.811	3.715	2.763	3.698	2.813	3.498
6	1.714×	0.286×	2.941	4.470	3.247	3.842	3.739	2.785	3.707	2.828	3.517
8	1.778×	0.222×	2.965	4.517	3.264	3.853	3.762	2.806	3.711	2.847	3.537

Table 11: Quality–efficiency trade-off across K at the 336M scale with $B = 64$ under matched compute. R_{view} and R_{unique} are defined relative to $K = 1$.

K	$\mu = 8$			$\mu = 16$		
	Step (s)	Speedup	Mem.	Step (s)	Speedup	Mem.
1	5.23	1.00×	66.0	OOM	–	OOM
2	4.02	1.30×	53.7	OOM	–	OOM
4	3.40	1.54×	46.4	3.26	1.60×	79.2
8	3.04	1.72×	43.3	2.99	1.75×	73.4

Table 12: Complete speed and memory results for the 1028M model at context length 2k on A800-80GB GPUs. Memory is reported in GiB.

Context	K	Mem. ($\mu = 4$)	Mem. ($\mu = 8$)
4k	1	64.5	OOM
	2	52.4	OOM
	4	46.1	79.0
8k	1	118.3	OOM
	2	91.8	168.2
	4	81.5	145.7

Table 13: Memory usage (GiB) at longer context lengths. 4k is measured on A800-80GB GPUs and 8k on B200-180GB GPUs.

and standard deviation across the three seeds. We also report a six-task average that excludes GPQA, which is close to chance at this scale. The complete per-seed results are reported in Table 15.

The independent runs show that $K = 2$ provides comparable downstream quality to $K = 1$.

D.3 Extended $K = 1$ Training

To assess whether 6,000 optimizer steps are sufficient for the $K = 1$ baseline to converge, we extend the original AR-to-Diffusion run to 10,000 steps. The endpoint of the cosine learning-rate schedule is extended accordingly, while all other training settings remain unchanged. For each training horizon, we report the mean and standard deviation across its final 10 consecutive checkpoints: steps 5,991–6,000 and 9,991–10,000, respectively. Because only one training seed is extended, the reported standard deviations characterize variation across adjacent checkpoints rather than run-to-run uncertainty.

Category	Proportion
chat	4.00%
code	40.00%
math	54.00%
ifeval	2.00%

Table 14: Normalized sampling mixture of the SFT data used for AR-to-Diffusion fine-tuning.

Extending training from 6,000 to 10,000 steps leaves the seven-task average essentially unchanged, from 48.72 to 48.62. These results suggest that the $K = 1$ baseline has largely plateaued under the evaluated AR-to-Diffusion training recipe.

E Artifact Licenses and Terms

We use publicly available datasets, model checkpoints, codebases, and benchmarks listed throughout the main paper and in Appendix B.1 and D.1 solely for academic research, training, and evaluation purposes. Our use is consistent with the intended use specified by each artifact’s original providers, and we follow their corresponding licenses and terms of use. We do not redistribute any of the original artifacts. Code and model checkpoints released by this work will be made available under a license compatible with all upstream artifacts used during their creation.

<i>K</i>	Seed	GSM8K	MATH	MMLU	GPQA	IFEval	MBPP	HumanEval	Avg.	Avg. w/o GPQA
1	32	70.34	41.70	51.96	25.96	59.87	49.73	40.93	48.64	52.42
	37	69.83	41.58	51.74	26.83	61.28	49.30	38.72	48.47	52.08
	42	69.70	41.02	51.69	25.22	62.36	47.43	43.60	48.72	52.63
	Mean $\pm$ Std.	69.96 $\pm$ 0.34	41.43 $\pm$ 0.36	51.80 $\pm$ 0.14	26.00 $\pm$ 0.81	61.17 $\pm$ 1.25	48.82 $\pm$ 1.22	41.08 $\pm$ 2.44	48.61 $\pm$ 0.13	52.38 $\pm$ 0.28
2	32	70.20	41.50	51.00	28.57	60.98	46.85	42.80	48.84	52.22
	37	69.35	40.80	51.66	26.47	59.45	50.54	42.13	48.63	52.32
	42	70.46	41.54	52.70	23.41	60.56	49.88	44.08	48.95	53.20
	Mean $\pm$ Std.	70.00 $\pm$ 0.58	41.28 $\pm$ 0.42	51.79 $\pm$ 0.86	26.15 $\pm$ 2.59	60.33 $\pm$ 0.79	49.09 $\pm$ 1.97	43.01 $\pm$ 0.99	48.81 $\pm$ 0.16	52.58 $\pm$ 0.54
4	32	68.51	40.60	51.31	27.66	59.30	46.61	42.06	48.01	51.40
	37	69.95	40.72	50.65	25.72	60.29	47.39	41.28	48.00	51.71
	42	69.87	40.87	51.61	27.46	59.85	44.98	43.11	48.25	51.72
	Mean $\pm$ Std.	69.44 $\pm$ 0.81	40.73 $\pm$ 0.14	51.19 $\pm$ 0.49	26.95 $\pm$ 1.07	59.81 $\pm$ 0.50	46.33 $\pm$ 1.23	42.15 $\pm$ 0.92	48.09 $\pm$ 0.14	51.61 $\pm$ 0.18

Table 15: Three-seed downstream results for AR-to-Diffusion adaptation initialized from Qwen2.5-1.5B-Instruct. Each seed-level entry is averaged over the final 10 checkpoints; the “Mean $\pm$ Std.” rows report the mean and standard deviation across seeds.

Steps	GSM8K	MATH	MMLU	GPQA	IFEval	MBPP	HumanEval	Avg.
6,000	69.70 $\pm$ 0.42	41.02 $\pm$ 0.27	51.69 $\pm$ 0.12	25.22 $\pm$ 0.43	62.36 $\pm$ 0.76	47.43 $\pm$ 0.67	43.60 $\pm$ 0.77	48.72
10,000	70.93 $\pm$ 0.48	42.24 $\pm$ 0.20	50.67 $\pm$ 0.06	26.72 $\pm$ 0.40	60.14 $\pm$ 0.97	45.99 $\pm$ 0.70	43.66 $\pm$ 0.72	48.62

Table 16: Downstream performance of the $K = 1$ AR-to-Diffusion baseline at 6,000 and 10,000 optimizer steps. Each entry reports the mean and checkpoint-wise standard deviation over the final 10 consecutive checkpoints.