dMoE: dLLMs with Learnable Block Experts

Sicheng Feng, Zigeng Chen, Gongfan Fang, Xinyin Ma, Xinchao Wang*
National University of Singapore
fengsicheng@u.nus.edu, xinchao@nus.edu.sg

Abstract

Diffusion Large Language Models (dLLMs) have recently emerged as a promising alternative to autoregressive models, offering competitive performance while naturally supporting parallel decoding. However, as dLLMs are increasingly integrated with Mixture-of-Experts (MoE) architectures to scale model capacity, a fundamental mismatch arises between block parallel decoding and token-level expert selection. Specifically, each dLLM forward pass processes multiple tokens with bidirectional dependencies, whereas conventional MoE layers route each token independently. This mismatch substantially increases the number of uniquely activated experts, making inference increasingly memory-bound. To address this, we propose dMoE, a simple yet effective block-level MoE framework. The central idea of dMoE is to aggregate token-level expert distributions within each block into a unified block-level expert distribution, which is then used to guide block expert routing in a coherent manner. In this way, dMoE substantially reduces the number of uniquely activated experts during inference without sacrificing performance, thereby mitigating the memory-bound bottleneck. Extensive experiments across a variety of benchmarks demonstrate the effectiveness of dMoE. On average, dMoE reduces the number of uniquely activated experts from 69.5 to 16.6 while retaining 99.55% of the original performance. Meanwhile, it reduces memory usage by 74.36% to 76.78% and achieves 1.36× to 2.04× end-to-end latency speedup. Code is available at: https://github.com/fscdc/dMoE.

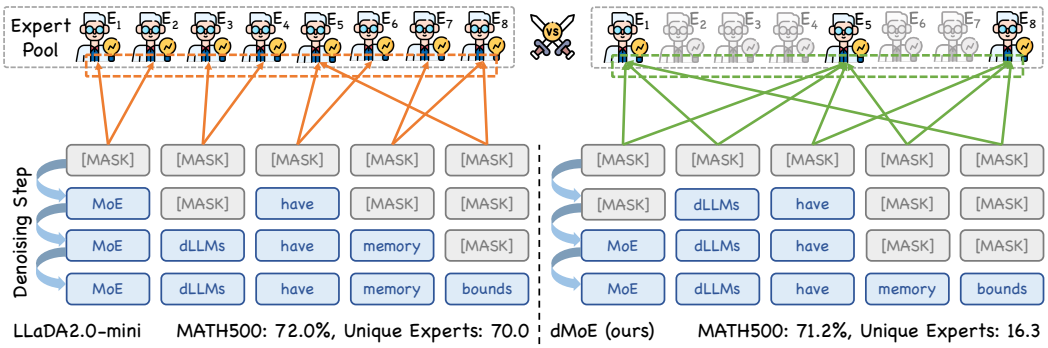


Figure 1: Comparison between the original LLaDA2.0-mini and our proposed dMoE. Unlike the original token-level expert routing in MoE dLLMs, our dMoE replaces token-level routing with block-level routing, substantially reducing the unique expert count while preserving performance.

1 Introduction

Recently, diffusion large language models (dLLMs) [1, 2, 3, 4, 5] have emerged as a competitive alternative to autoregressive LLMs [6, 7, 8], demonstrating strong performance in both open-source

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and closed-source settings [9, 10]. By iteratively refining mask tokens through an unmask-and-remask process, dLLMs naturally support parallel decoding beyond the left-to-right generation order of autoregressive models, offering greater flexibility and efficiency potential at test time. To further scale model capacity while keeping the number of active parameters manageable, recent dLLMs [11, 12, 13, 14, 15, 16] have increasingly been integrated with Mixture-of-Experts (MoE) architectures, making MoE a growing design trend in this paradigm.

While MoE architectures offer an effective scaling strategy by increasing model capacity with sparse activation, they also introduce a fundamental efficiency challenge. In MoE dLLMs, existing MoE routing still follows the token-level expert selection paradigm inherited from autoregressive models, selecting experts independently for each token. However, dLLMs process multiple tokens within a single forward pass (*e.g.*, a whole block of tokens in block diffusion decoding [17]). As a result, the number of uniquely activated experts can grow dramatically within one forward pass, making memory access the primary inference bottleneck. The empirical results in Section 3 further support our claims by showing that MoE latency dominates the end-to-end inference latency and linearly increases with the number of uniquely activated experts.

A large body of prior work has studied efficient MoE strategies, primarily in autoregressive models. Existing methods can be broadly divided into two categories: pre-execution compression, such as expert pruning [18, 19, 20, 21, 22] and expert merging [23, 24, 25], and runtime-adaptive execution, such as expert skipping [26, 27, 28], adaptive expert selection [29], and expert reuse [30, 31]. However, MoE efficiency in dLLMs remains largely underexplored. Unlike autoregressive decoding, dLLMs perform parallel token generation and refinement within each denoising step, making expert activation patterns and efficiency bottlenecks fundamentally different. A few recent studies have begun to explore this setting. EC-DLM [32] replaces token-choice routing with expert-choice routing and improves load balancing by dynamically adjusting expert capacity. TEAM [33] leverages temporal and spatial consistency in routing to reuse experts across denoising steps, thereby reducing overall expert activation during inference. DES [34] explicitly targets the memory overhead caused by excessive expert activation through a candidate-constrained routing strategy.

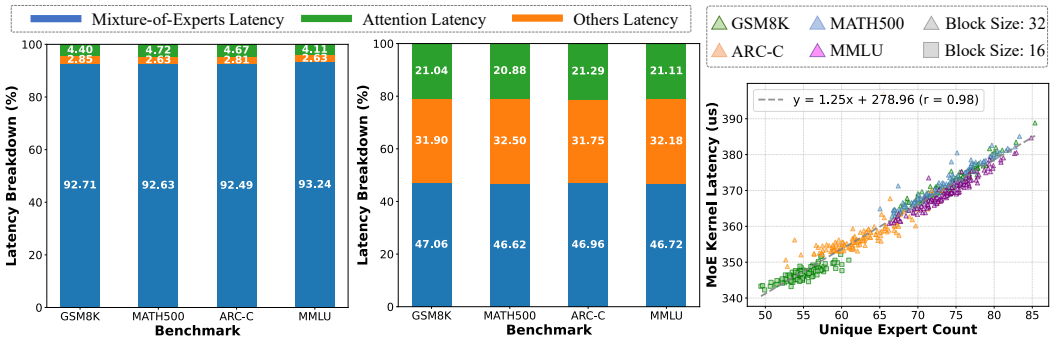
We propose dMoE, a simple yet effective strategy for compressing unique experts in MoE dLLMs. Our design is motivated by two key observations. First, token-level expert scores provide an informative signal of expert importance. Second, the degree of expert concentration varies substantially across denoising steps and blocks. Specifically, we first aggregate token-level expert scores to form block-level expert scores, and then use these block-level scores to guide the original routing process, thereby controlling the number of uniquely activated experts. In this way, dMoE can aggressively reduce the unique expert count without changing the number of experts selected for each token. Moreover, dMoE dynamically controls the unique expert count with a top-p criterion, allowing it to better adapt to the varying routing characteristics across different denoising steps and blocks. During training, we adopt a self-distillation paradigm using the same routing procedure in the forward pass.

We choose LLaDA2.0-mini, a popular open-source dLLM, as the base model for fine-tuning and evaluation. We evaluate dMoE on six benchmarks, including MATH500 [35], GSM8K [36], ARC-C [37], MMLU [38], TruthfulQA [39], and QASPER [40]. The results show that dMoE consistently achieves substantial expert compression with no performance degradation (Figure 1). On average, dMoE reduces the number of uniquely activated experts by $4.19\times$ while retaining 99.55% of the original performance. In addition, it reduces memory usage by 74.36% to 76.78% and delivers $1.36\times$ to $2.04\times$ end-to-end latency speedup compared with the original model. Our dMoE also achieves a superior performance-efficiency trade-off compared with the baselines. Furthermore, our dMoE is tunable, allowing the number of activated experts to be adjusted to different application requirements.

Overall, we introduce dMoE, a novel learnable strategy for block-level expert routing in MoE dLLMs. The core idea of dMoE is to aggregate token-level expert scores into block-level expert scores, and then use these block-level scores to dynamically guide the original routing process. Extensive experiments demonstrate the effectiveness of our method. This work establishes a strong baseline for block-level routing in MoE dLLMs.

2 Related Work

Overview of Diffusion Language Models. Diffusion-based generative modeling has achieved remarkable success in continuous modalities, including images [41, 42], videos [43, 44], and au-



(a) Latency Breakdown (Transformers) (b) Latency Breakdown (SGLang) (c) MoE Kernel Latency (SGLang)

Figure 2: Empirical studies on LLaDA2.0-mini. (a) & (b) We report the latency breakdown of three components: MoE (*e.g.*, routing and execution), attention (*e.g.*, attention computation and related projections), and others (*e.g.*, LM head and embeddings). (c) We present the relationship between the number of unique activated experts and MoE kernel latency.

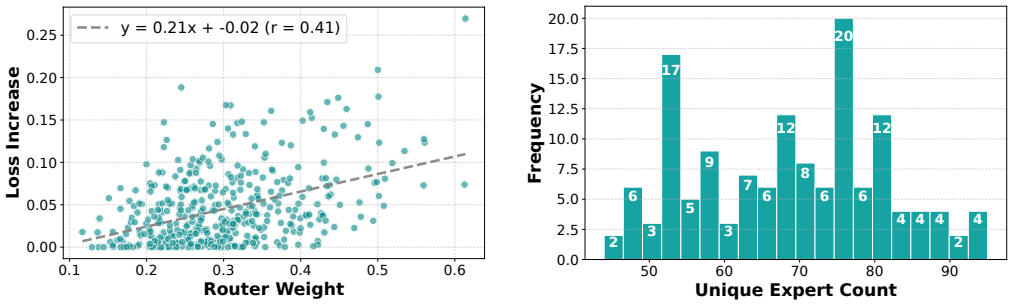
dio [45, 46], building on the broader foundation of diffusion models [47, 48, 49]. Extending this framework to language, however, is non-trivial because text is inherently discrete. To address this challenge, a growing body of work formulates diffusion directly in token space [50, 51, 52, 53, 14, 3, 54, 55, 56], often through masked-token denoising, which enables parallel generation and relaxes the strict left-to-right dependency of autoregressive decoding. Based on this formulation, dLLMs [3, 4, 10, 9, 11] have shown increasingly competitive performance at the billion-parameter scale, suggesting that diffusion is becoming a practical alternative for language generation. Recent progress also suggests a clear trend toward sparse scaling in dLLMs, with an increasing number of representative models adopting MoE architectures to expand overall model capacity while keeping the number of active parameters at each denoising step relatively small [13, 11, 12]. Beyond general text generation, diffusion-based language modeling is now being extended to more challenging settings, including reasoning [57, 58, 59, 60, 61, 62], multimodal generation [63, 64, 5, 65], and code synthesis [66, 10, 67], highlighting the rapid expansion and growing maturity of this research direction [68, 69, 70].

Efficient Mixture-of-Expert Strategies. In autoregressive models, most existing studies in this setting are designed to reduce token-wise expert computation or serving overhead, and are therefore primarily tailored to left-to-right decoding. Broadly, these methods can be grouped into two lines. The first line focuses on pre-execution compression, which reduces the model-side budget before inference, for example, through expert pruning [18, 19, 20, 21, 22] and expert merging [23, 24, 25]. The second line focuses on runtime-adaptive execution, which improves efficiency during inference by dynamically controlling expert activation according to the current input, such as expert skipping [26, 27, 28], adaptive expert selection [29], and expert reuse [30, 31]. Beyond these algorithmic strategies, another important direction lies in system-level optimization [71, 72], which improves MoE efficiency from the perspectives of communication and memory.

However, MoE efficiency in dLLMs remains largely underexplored. Some recent studies have provided initial evidence showing its potential. EC-DLM [32] replaces conventional token-choice routing with expert-choice routing and further improves load balancing by dynamically adjusting expert capacity across denoising steps. TEAM [33] exploits temporal and spatial consistency in expert routing to reuse experts across denoising steps, thereby reducing the overall number of activated experts during inference. DES [34] explicitly targets the memory overhead caused by excessive expert activation within each block under parallel decoding. It introduces a candidate-constrained routing strategy to compress the number of block-level activated experts without any additional training. Our target is closely aligned with DES, but pushes this direction further by pursuing more aggressive block-level expert selection to achieve stronger expert compression.

3 Preliminaries

We briefly introduce dLLMs from the perspective most relevant to this work: their native support for parallel decoding. Unlike autoregressive language models, which generate tokens strictly from left to right, dLLMs predict multiple unknown positions simultaneously by iteratively refining a partially masked sequence. This property makes them especially suitable for parallel generation, since the



(a) Router weight *vs.* Loss increase

(b) Distribution of unique expert count

Figure 3: (a) Router weights of the masked experts (x-axis) *vs.* the resulting loss increase (y-axis) at Layer 10 (the middle layer), with 432 data points sampled across denoising steps on GSM8K. (b) Distribution of the unique expert count during inference on GSM8K (details in Appendix A.1).

model can update a set of masked tokens in one denoising step rather than committing to a single next token at a time. A common instantiation is the masked diffusion language model (MDLM), where generation starts from a corrupted sequence x_t obtained by masking tokens from a clean sequence x_0. Let $t \in [0, 1]$ denote the masking intensity. The forward corruption process independently replaces each token with [MASK] according to

$$q(x_t \mid x_0) = \prod_{i=1}^{L} \left[(1-t)\,\delta(x_t^i = x_0^i) + t\,\delta(x_t^i = \text{[MASK]}) \right]. \tag{1}$$

Given the corrupted sequence, the denoising model p_θ predicts the original tokens at the masked positions:

$$p_\theta(x_0 \mid x_t) = \prod_{i:x_t^i=\text{[MASK]}} p_\theta(x_0^i \mid x_t). \tag{2}$$

Because all masked positions can be reconstructed in parallel within each denoising step, dLLMs naturally enable non-autoregressive parallel decoding.

Memory Bound Becomes the Primary Bottleneck for Mixture-of-Experts dLLMs. In MoE dLLMs, each forward pass needs to select experts for multiple tokens simultaneously (*e.g.*, a full block of tokens under block diffusion decoding [17]). As a result, a large number of unique experts can be activated within a single forward pass, and these experts need to be iteratively loaded during inference, leading to substantial memory overhead. We further conduct empirical studies on LLaDA2.0-mini across various benchmarks to support this claim. Specifically, for all empirical studies, we use block diffusion decoding and include a warm-up stage before measurement to stabilize memory, cache, and kernel states. As shown in Figure 2 (a) and (b), we report the latency breakdown of three components: MoE, attention, and others. Here, MoE includes routing and expert execution, attention includes attention computation and related projections, and others include components such as the LM head and embeddings. We report results under both the native Transformers[2] and SGLang[3] frameworks, and consistently observe that MoE latency dominates the end-to-end latency. In Figure 2 (c), we additionally record the MoE kernel latency together with the corresponding unique expert count, *i.e.*, the number of uniquely activated experts within a single forward pass; under block diffusion, this corresponds to the unique experts activated by all tokens in the current block. The results show a clear linear positive correlation, indicating that activating more unique experts leads to higher MoE kernel latency. Overall, these findings motivate us to alleviate the memory bottleneck in MoE dLLMs by reducing the number of unique activated experts.

4 Methods

Based on the analysis in Section 3, we focus on reducing the number of block-level unique experts to alleviate the memory bottleneck while keeping the computation unchanged, that is, maintaining a fixed number of selected experts for each token. In this section, we first present two key observations in Section 4.1, and then introduce our proposed dMoE in detail in Section 4.2.

[2] https://github.com/huggingface/transformers
[3] https://github.com/sgl-project/sglang

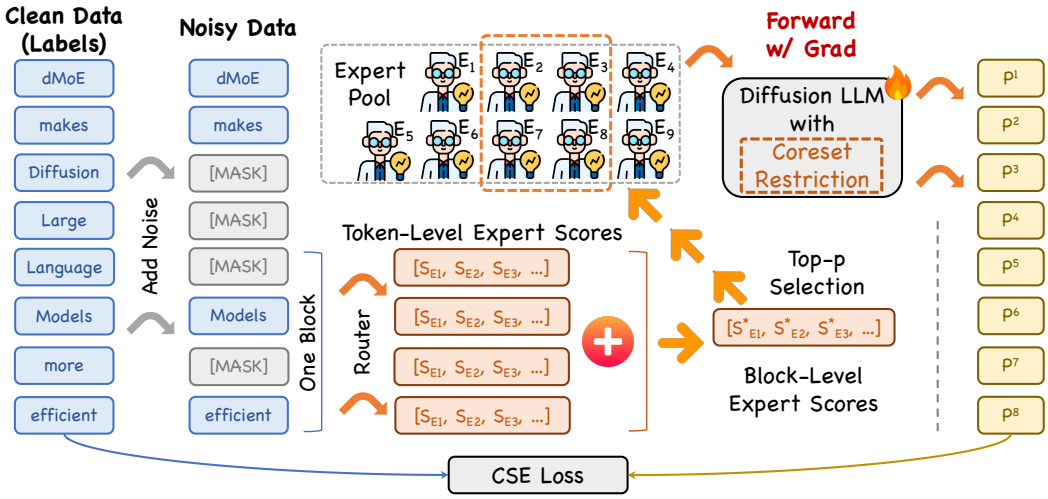


Figure 4: Overview of our proposed dMoE. For each noisy block, we aggregate token-level router scores into block-level expert scores and apply a top-p criterion to select an adaptive expert coreset. The final token-level routing is then restricted to this coreset. Training follows a self-distillation paradigm using CSE loss, and inference uses the same forward process.

4.1 Key Observations

We conduct empirical studies of block diffusion decoding on LLaDA2.0-mini. From these results, we identify several key observations that motivate the design of our dMoE for expert compression in MoE dLLMs.

Observation A. *Token-level expert scores provide an effective indicator of expert importance.* To verify this, we analyze the relationship between token-level expert scores (*i.e.*, router weights) and the change in reconstruction loss after masking the corresponding experts. Specifically, we randomly sample 432 data points from GSM8K, using the original LLaDA2.0-mini as the base model, and compute the correlation between these two quantities. As shown in Figure 3 (a), they exhibit a clear positive correlation, with a Pearson correlation coefficient of 0.41. This observation motivates us to aggregate token-level expert scores for block-level expert selection.

Observation B. *The degree of expert concentration varies substantially across denoising steps and blocks.* To examine this phenomenon in more detail, we further analyze the distribution of the unique expert count for the generated block at Layer 10 during inference, where Layer 10 corresponds to the middle layer of LLaDA2.0-mini, which contains 19 MoE layers in total. As illustrated in Figure 3 (b), the unique expert count changes noticeably across different denoising steps, suggesting that the extent to which tokens are routed to a concentrated subset of experts is highly step-dependent. To further quantify this variability, we adopt the coefficient of variation (CV), defined as

Table 1: Results of the coefficient of variation (CV) on various benchmarks.

Metric	GSM8K	MATH500	ARC-C	MMLU
Mean	70.01	71.09	62.85	66.57
Std	11.96	11.34	10.01	11.85
CV	17.09%	15.95%	15.93%	17.80%

$$\mathrm{CV} = \frac{\sigma}{\mu},$$

where σ and μ denote the standard deviation and mean of the unique expert count, respectively. As shown in Table 1, the unique expert count exhibits consistently large variation across various benchmarks, suggesting that this phenomenon is not tied to a specific dataset or task. These results provide further empirical evidence that expert concentration varies substantially across denoising steps and blocks.

4.2 dMoE

To achieve our goal of reducing the number of block-level unique experts, a natural idea is to constrain the expert pool during expert selection. Following a coarse-to-fine strategy, we first select a

coreset from the full expert pool, and then perform token-level expert selection within this coreset (Figure 4). Specifically, given the tokens within a block, we first perform token-level routing to compute token-level expert scores:

$$\mathbf{s}_i = \text{Router}(t_i) = \left[s_{E_1},\, s_{E_2},\, \ldots,\, s_{E_{|\mathcal{E}|}}\right],$$

where t_i denotes the representation of the i-th token in the block, and s_{E_i} is the routing score of expert E_i for token i. Inspired by observation A, which suggests that token-level expert scores are positively correlated with expert importance, we then directly aggregate the token-level expert scores to obtain block-level expert scores:

$$S_{block} = \oplus_{i \in \mathcal{B}} s_i,$$

where $\mathcal{B}$ denotes the set of tokens in the current block, and $\oplus$ represents the aggregation operator. After obtaining the block-level expert scores, we further determine the coreset based on their normalized values. Inspired by observation B, which suggests that the degree of expert concentration varies substantially across denoising steps and blocks, we do not enforce a fixed coreset size. Instead, we first normalize the block-level expert scores and then apply a top-p criterion to select the coreset:

$$\mathcal{C} = \text{top-p}(\{\tilde{S}_{block}\}_{e=1}^{E},\, p),$$

where $\mathcal{C}$ denotes the selected coreset and p is the cumulative probability threshold. This design naturally yields a smaller coreset when the expert score distribution is concentrated, and a larger one when the distribution is more dispersed, thereby adapting to the routing characteristics of different blocks. As such, it is better aligned with the original inference behavior. Finally, the original token-level routing is performed within the selected coreset (see more details in Algorithm 1).

Algorithm 1: Block-Level Expert Routing

Input: Token-level expert scores $\{\mathbf{s}_i\}_{i \in \mathcal{B}}$
Output: Block-level expert scores S_{block}, coreset $\mathcal{C}$, routed experts $\{R_i\}_{i \in \mathcal{B}}$

Initialize $S_{block} \leftarrow \mathbf{0} \in \mathbb{R}^{|\mathcal{E}|}$;
foreach $i \in \mathcal{B}$ **do**
  $\hat{\mathbf{s}}_i \leftarrow \text{TopKMask}(\mathbf{s}_i, k)$;
  $S_{block} \leftarrow S_{block} + \hat{\mathbf{s}}_i$;
$\tilde{S}_{block} \leftarrow \text{Normalize}(S_{block})$;
$\mathcal{C} \leftarrow \text{top-p}(\tilde{S}_{block}, p)$;
foreach $i \in \mathcal{B}$ **do**
  $\mathbf{s}_i^{\mathcal{C}} \leftarrow \text{MaskOutside}(\mathbf{s}_i, \mathcal{C})$;
  $R_i \leftarrow \text{TopK}(\mathbf{s}_i^{\mathcal{C}}, k)$;
return $S_{block}, \mathcal{C}, \{R_i\}_{i \in \mathcal{B}}$;

During the training process, we adopt a self-distillation paradigm [73] and follow the above routing procedure in the forward pass. During the inference process, we use the same forward process to maintain alignment between training and inference.

5 Experiments

5.1 Experimental Setup

Training Data. All training data are constructed via self-distillation [73]. We first collect prompts from several public datasets, including the GSM8K training set [36], PRM12K [35], a subset of Numina-Math [74], and a subset of OpenThoughts [75]. We then use LLaDA2.0-mini (*i.e.*, our base model) to generate corresponding responses as supervision targets. During generation, we follow the official settings: the confidence threshold is set to 0.95, the block size to 32, and the maximum output length to 2048 tokens. Samples that fail to terminate within this length limit are removed. Importantly, the entire supervision signal is derived from the model's own generations, without relying on any externally curated high-quality responses. Overall, we obtain approximately 700K training samples.

Training Details. We use LLaDA2.0-mini as the base model, which is a popular open-source MoE dLLM. During training, we randomly sample the masking ratio from $[0.3, 0.8]$ for each training instance. We perform full fine-tuning for 2 epochs with a global batch size of 4. The learning rate is set to 2.0×10^{-6} with a cosine learning rate schedule. We adopt the block diffusion setting and follow the official configuration by setting the block size to 32. We train two models with the top-p threshold p^{train} set to 0.6 (default) and 0.5, respectively. All training is conducted on 4 H100 GPUs.

Baselines. We compare dMoE against six baselines: (1) **Original**, which follows the official configuration with block diffusion inference; (2) **+ Fine-Tuning**, which keeps the original routing but applies the same self-distillation fine-tuning as dMoE; (3) **Top-4**, which reduces the number of selected experts per token from 8 in the official setting to 4; (4) **DES-S**, where we implement

Table 2: Results on various benchmarks with LLaDA2.0-mini. For MMLU, we use the math part for evaluation. Since the official DES code is not publicly available, we implement DES for comparison. Expert count denotes the average number of unique activated experts per layer. $*$ indicates that the corresponding threshold in DES is adjusted to achieve a higher compression ratio.

Method	MATH500		GSM8K		ARC-C		MMLU	
	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy
LLaDA2.0-mini	70.0	72.0%	71.5	90.8%	65.9	84.4%	70.6	88.6%
Baselines								
+ Fine-Tuning	79.1	72.2%	79.7	91.0%	70.7	87.0%	78.9	89.2%
Top-4	43.5	52.6%	44.3	72.4%	42.2	74.0%	43.8	61.5%
DES-S	41.0	70.8%	41.9	90.7%	39.9	83.3%	41.4	87.1%
DES-V	36.9	71.8%	37.0	90.5%	36.8	82.3%	37.0	85.6%
DES-S*	12.9	53.8%	13.2	74.0%	13.0	62.9%	13.0	70.7%
DES-V*	13.0	53.0%	12.9	73.5%	13.0	49.1%	13.0	72.6%
TEAM	43.6	71.2%	44.8	90.1%	38.4	85.0%	43.8	87.4%
dMoE								
ours ($p^{train} = p^{test} = 0.6$)	16.3	71.2%	16.6	88.6%	16.9	87.1%	16.5	87.4%
ours ($p^{train} = p^{test} = 0.5$)	13.7	69.8%	13.8	87.1%	13.9	84.7%	13.6	86.7%

DES-Seq [34], a sequence-level routing strategy that enables more adaptive expert allocation; (5) **DES-V**, where we implement DES-Vote [34], which reduces the number of unique experts within each block by consolidating router-derived preferences across tokens; and (6) **TEAM** [33], which reuses experts across denoising steps by exploiting the temporal and spatial consistency in expert routing. Additionally, for both DES-S and DES-V, the degree of expert compression can be controlled through hyperparameters.

Inference Details. We follow the official inference settings to evaluate the LLaDA2.0-mini [11]. For the baselines, we report the results from our implementation, with the latter following the configurations specified in the original paper. We apply the block diffusion setting with confidence-based parallel decoding, consistent with the official implementation. We follow the official settings for both our dMoE and all the baselines: set block size to 32, set max generation length to 2048, set confidence threshold to 0.95, and enable early stopping. All results in this paper are obtained with KV caching enabled for all methods. Unless otherwise specified, we use a batch size of 1 for inference, and results with larger batch sizes are provided in Table 5.

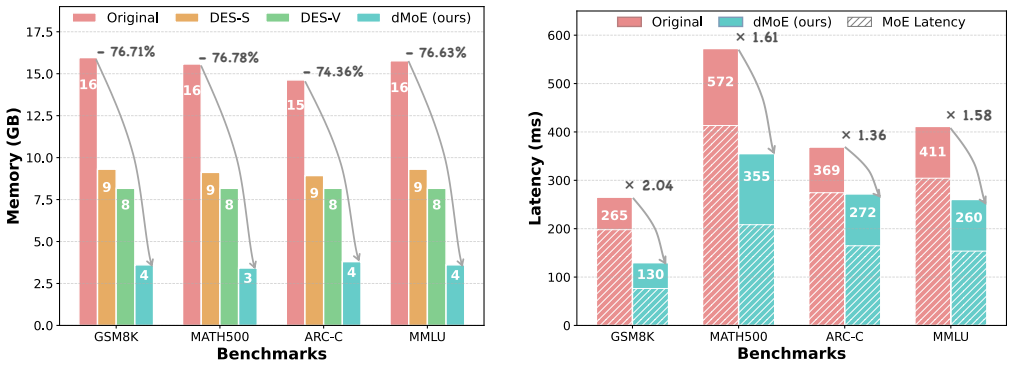
Evaluation Details. We conduct comprehensive experiments on multiple benchmarks spanning a broad spectrum of reasoning tasks, including mathematical reasoning, scientific reasoning, and general high-level reasoning. Specifically, we evaluate on MATH500 [35], GSM8K [36], ARC-C [37], and MMLU [38]. In addition, we follow the simple-eval framework[4] for zero-shot evaluation and prompt the model to generate its reasoning trajectories step by step. Beyond reasoning tasks, we further evaluate on two general benchmarks, TruthfulQA [39] and QASPER [40].

5.2 Main Results

dMoE Achieves Aggressive Expert Compression While Preserving Performance. We report the main results in Table 2. Compared with the original model, our dMoE achieves substantial expert compression while preserving performance almost intact. Specifically, the average performance across four benchmarks decreases only slightly from 83.95% to 83.575%, retaining 99.55% of the original performance, while reducing the average unique expert count from 69.5 to 16.575, corresponding to a 76.15% reduction in the average unique expert count. Compared with the baselines, our dMoE further achieves a 55.11% to 61.14% greater reduction in unique expert count at a comparable performance level. Moreover, under the same self-distillation fine-tuning, dMoE retains 98.50% of the performance of + Fine-Tuning with 78.50% fewer unique experts, indicating that the expert compression stems from block-level routing rather than fine-tuning.

dMoE Generalizes Beyond Reasoning Tasks. We further report the results on two general benchmarks in Table 3. dMoE reduces the average unique expert count from 61.25 to 15.4, corresponding to a 74.86% reduction, while maintaining comparable overall performance. Notably, dMoE improves

[4] https://github.com/openai/simple-evals



(a) Memory comparison

(b) Latency speedup

Figure 5: (a) We report the average memory footprint of uniquely activated MoE parameters across all layers for both the baselines and our dMoE. (b) We present the average end-to-end latency and MoE latency for the original model and our dMoE.

Table 4: Latency comparison with DES on LLaDA2.0-mini. We report the average end-to-end latency (ms) and the speedup over the original model. All methods are compared at comparable accuracy.

Method	MATH500		GSM8K		ARC-C		MMLU	
	Latency (ms)	Speedup	Latency (ms)	Speedup	Latency (ms)	Speedup	Latency (ms)	Speedup
LLaDA2.0-mini	572.4	1.00×	265.1	1.00×	368.7	1.00×	411.4	1.00×
DES-S	394.2	1.45×	161.4	1.64×	300.4	1.22×	312.8	1.32×
DES-V	381.6	1.50×	187.3	1.42×	290.5	1.27×	300.5	1.37×
ours	355.1	1.61×	129.7	2.04×	271.8	1.36×	260.4	1.58×

the accuracy on TruthfulQA from 62.2% to 69.9%, with only a slight drop on QASPER from 66.5% to 64.8%. These results demonstrate that the effectiveness of dMoE extends beyond reasoning tasks.

dMoE Significantly Reduces Memory Usage. As shown in Figure 5 (a), we compare the memory footprint of the original model, the baselines, and our dMoE. Specifically, we report the average memory footprint of uniquely activated MoE parameters across all layers. The results show that our dMoE achieves the lowest memory usage, yielding a 74.36% to 76.78% reduction compared with the original model and a 54.08% to 60.38% reduction compared with the baselines.

Table 3: Results on general benchmarks. For TruthfulQA, we report BLEU accuracy. EC denotes the average number of unique activated experts.

Method	TruthfulQA		QASPER	
	EC	Accuracy	EC	Accuracy
LLaDA2.0-mini	61.2	62.2%	61.3	66.5%
ours	15.4	69.9%	15.4	64.8%

dMoE Brings End-to-End Speedup. We further report the end-to-end latency (including MoE latency), together with the corresponding speedup, in Figure 5 (b) to evaluate the practical efficiency of our method. Across four benchmarks, our method achieves 1.36× to 2.04× end-to-end latency speedup, with the gain mainly coming from the reduction in MoE latency. In addition, we compare the latency with DES in Table 4, where all methods are compared at comparable accuracy. Our dMoE achieves the highest speedup on all four benchmarks, whereas DES-S and DES-V only achieve 1.22× to 1.64× and 1.27× to 1.50× speedup, respectively. Moreover, the overhead of top-p selection and coreset construction is negligible, accounting for only 1.9% of the total MoE latency on average. These results demonstrate the effectiveness of our method in delivering real inference acceleration.

dMoE Performs Better at Extreme Expert Compression. We further adjust the thresholds of DES-S and DES-V to achieve compression ratios comparable to those of our dMoE (DES-S* and DES-V* in Table 2). We can observe that under similar compression levels, our dMoE remains nearly lossless, retaining 99.55% of the original performance with $p = 0.6$ and 97.77% with $p = 0.5$, whereas DES-S and DES-V suffer from substantial performance degradation. Notably, our dMoE can compress the number of unique experts to near the practical limit, *i.e.*, the number of experts required per token in the model, which is 8 in LLaDA2.0-mini, while still maintaining strong performance.

dMoE Remains Effective with Larger Batch Sizes. Since larger batch sizes may activate more experts, we further evaluate dMoE with batch sizes of 4 and 8. As shown in Table 5, dMoE achieves

Table 5: Results with larger batch sizes on LLaDA2.0-mini. We report the average latency (Lat., ms), the speedup over the original model (Spd.), the unique expert count across the entire batch (EC), and the memory footprint of uniquely activated MoE parameters (M, GB).

Method	MATH500				GSM8K				ARC-C				MMLU			
	Lat.	Spd.	EC	M	Lat.	Spd.	EC	M	Lat.	Spd.	EC	M	Lat.	Spd.	EC	M
Batch size $= 4$																
LLaDA2.0-mini	349.9	1.00×	117.1	27	167.6	1.00×	117.5	27	169.7	1.00×	113.4	26	297.5	1.00×	109.5	25
ours	227.3	1.54×	34.4	8	91.4	1.83×	36.6	8	98.8	1.72×	38.5	9	201.9	1.47×	33.1	8
Batch size $= 8$																
LLaDA2.0-mini	292.9	1.00×	136.4	31	124.8	1.00×	135.3	31	152.3	1.00×	118.9	27	212.3	1.00×	132.9	30
ours	188.0	1.56×	44.3	10	67.7	1.84×	49.1	11	104.8	1.45×	50.9	12	149.4	1.42×	44.6	10

Table 6: Ablation on the cumulative probability threshold p at training and test stages. We evaluate cumulative probability thresholds of 0.4, 0.5, 0.6, 0.7, and 0.8 while keeping other settings fixed.

Method	MATH500		GSM8K		ARC-C		MMLU	
	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy
LLaDA2.0-mini	70.0	72.0%	71.5	90.8%	65.9	84.4%	70.6	88.6%
dMoE ($p^{train} = 0.6$)								
ours ($p^{test} = 0.4$)	13.7	67.0%	13.9	86.0%	13.1	83.7%	13.5	84.1%
ours ($p^{test} = 0.5$)	14.0	68.8%	14.2	87.5%	13.8	85.5%	13.7	87.0%
ours ($p^{test} = 0.6$)	16.3	71.2%	16.6	88.6%	16.9	87.1%	16.5	87.4%
ours ($p^{test} = 0.7$)	19.8	73.0%	20.1	90.2%	20.0	88.1%	20.2	87.4%
ours ($p^{test} = 0.8$)	27.0	73.2%	28.3	90.0%	28.0	89.2%	27.6	88.9%
dMoE ($p^{train} = 0.5$)								
ours ($p^{test} = 0.4$)	13.0	67.2%	13.5	85.9%	12.9	83.8%	13.1	84.1%
ours ($p^{test} = 0.5$)	13.7	69.8%	13.8	87.1%	13.9	84.7%	13.6	86.7%
ours ($p^{test} = 0.6$)	16.4	70.0%	16.5	88.3%	16.1	87.0%	16.5	87.0%
ours ($p^{test} = 0.7$)	19.7	73.2%	20.2	90.2%	20.1	88.4%	19.9	87.4%
ours ($p^{test} = 0.8$)	27.1	72.8%	28.1	90.4%	28.0	89.0%	27.5	88.1%

$1.47\times$ to $1.83\times$ and $1.42\times$ to $1.84\times$ speedup with batch sizes of 4 and 8, respectively, while reducing the unique expert count across the entire batch by 66.05% to 70.62% and 57.19% to 67.52%. These results further demonstrate the effectiveness and generalizability of dMoE.

5.3 Diagnostic Results

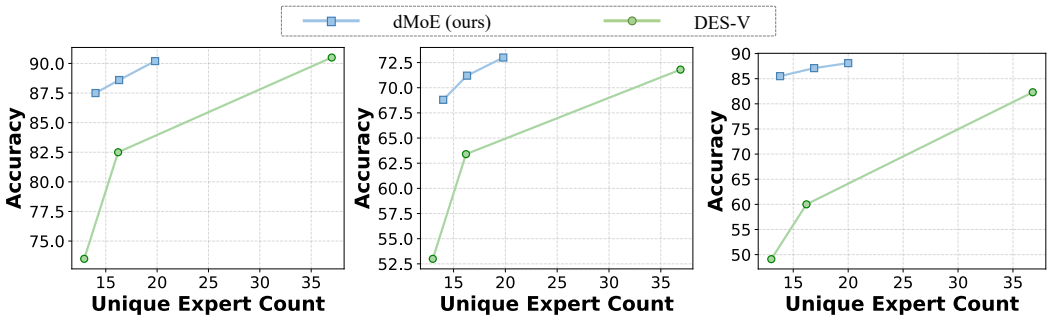
We provide comprehensive ablation studies on the cumulative probability threshold p and block size. Furthermore, we show that our dMoE achieves a superior performance-efficiency trade-off compared with the baselines.

Ablation on the Cumulative Probability Threshold p. We first conduct an ablation study on the cumulative probability threshold p. Specifically, we train two models with $p^{train} = 0.6$ and $p^{train} = 0.5$, respectively, and evaluate each tuned model under $p^{test} \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$. As shown in Table 6, when evaluated with the same p^{test}, the two models trained with different p^{train} achieve similar performance, while still maintaining very low unique expert counts. Moreover, as p^{test} increases, the unique expert count gradually rises, accompanied by improved performance. Overall, these results demonstrate not only the robustness of our method but also its tunability, allowing the number of activated experts to be flexibly adjusted to different hardware characteristics and application requirements.

Ablation on the Block Size. We further conduct an ablation study on the block size, with the results summarized in Table 7. Specifically, we evaluate the model under block sizes of 8, 16, 24, and 32. The results show that our dMoE consistently compresses the unique expert count while maintaining strong performance across different block sizes. Specifically, under block sizes of 32, 24, 16, and 8, our dMoE reduces the average unique expert count by 76.15%, 72.97%, 68.46%, and 53.97%, while retaining 99.55%, 98.62%, 99.79%, and 98.81% of the original performance, respectively. The consistent efficiency improvement without a performance drop across different block sizes further demonstrates the effectiveness of our method.

Table 7: Ablation on the block size. The evaluated model is trained with a block size of 32 with $p^{train} = 0.6$. We set $p^{test} = 0.6$ here. We evaluate block sizes of 8, 16, 24, and 32 while keeping all other hyperparameters fixed.

Method	MATH500		GSM8K		ARC-C		MMLU	
	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy	Expert Count	Accuracy
Block size = 32								
LLaDA2.0-mini	70.0	72.0%	71.5	90.8%	65.9	84.4%	70.6	88.6%
ours	16.3	71.2%	16.6	88.6%	16.9	87.1%	16.5	87.4%
Block size = 24								
LLaDA2.0-mini	60.7	73.6%	62.2	91.0%	57.8	82.3%	61.6	86.9%
ours	16.2	68.6%	17.4	87.4%	16.1	87.9%	15.8	85.3%
Block size = 16								
LLaDA2.0-mini	48.6	70.2%	49.8	90.5%	47.0	86.9%	49.6	88.5%
ours	14.8	70.0%	16.7	89.9%	15.1	88.1%	14.9	87.4%
Block size = 8								
LLaDA2.0-mini	31.5	71.0%	32.2	90.1%	31.1	87.7%	32.3	87.4%
ours	14.1	70.2%	16.1	88.1%	14.2	87.2%	14.1	86.7%



(a) GSM8K (b) MATH500 (c) ARC-C

Figure 6: Comparison of the performance-efficiency trade-off between our method and baselines. We report the results of GSM8K (a), MATH500 (b), and ARC-C (c).

dMoE Reaches Superior Performance-Efficiency Trade-off. We present the performance–efficiency trade-off between our dMoE and the baselines. The results of our dMoE are taken from Table 6. For DES, we compare against its stronger variant, DES-V, and obtain its trade-off points by varying the threshold. As shown in Figure 6, our dMoE achieves better performance with fewer unique experts, leading to a superior performance–efficiency trade-off.

6 Conclusion

In this work, we introduce dMoE, a novel strategy for block-level expert routing in MoE dLLMs. dMoE enables aggressive expert compression while incurring almost no performance degradation. Comprehensive evaluations across diverse benchmarks demonstrate the effectiveness of our method. More broadly, this work lays a foundation for block-level routing in MoE dLLMs and opens up a promising direction for improving their efficiency.

Acknowledgements

This project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2026, WBS: A-8004345-00-00), the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office (Award: CRPO-GC1-NTU-002), as well as NUS IT's High Performance Cluster.

References

[1] Qiuhua Yi, Xiangfan Chen, Chenwei Zhang, Zehai Zhou, Linan Zhu, and Xiangjie Kong. Diffusion models in text generation: a survey. *PeerJ Computer Science*, 2024.

[2] Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S Yu, et al. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*, 2025.

[3] 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. *arXiv preprint arXiv:2502.09992*, 2025.

[4] 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.

[5] Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion parallel generation for large multimodal modal. *Conference on Empirical Methods in Natural Language Processing*, 2026.

[6] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[7] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[9] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

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

[11] 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.

[12] Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Junlin Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. Llada2.1: Speeding up text diffusion via token editing, 2026.

[13] Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025.

[14] Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[15] Jinjie Ni and team. Openmoe 2: Sparse diffusion language models. https://github.com/JinjieNi/OpenMoE2, 2025.

[16] Haokun Lin, Xinle Jia, Shaozhen Liu, Shujun Xia, Weitao Huang, Haobo Xu, Junyang Li, Yicheng Xiao, Xingrun Xing, Ziyu Guo, et al. Efficient diffusion language models: A comprehensive survey. *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026.

[17] 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.

[18] Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.

[19] Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*, 2022.

[20] Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. *arXiv preprint arXiv:2405.16646*, 2024.

[21] Hongcheng Guo, Juntao Yao, Boyang Wang, Junjia Du, Shaosheng Cao, Donglin Di, Shun Zhang, and Zhoujun Li. Cluster-driven expert pruning for mixture-of-experts large language models. *arXiv preprint arXiv:2504.07807*, 2025.

[22] Chenyang Song, Weilin Zhao, Xu Han, Chaojun Xiao, Yingfa Chen, Yuxuan Li, Zhiyuan Liu, and Maosong Sun. Blockffn: Towards end-side acceleration-friendly mixture-of-experts with chunk-level activation sparsity. *arXiv preprint arXiv:2507.08771*, 2025.

[23] Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. Merging experts into one: Improving computational efficiency of mixture of experts. In *EMNLP*, 2023.

[24] Sejik Park. Learning more generalized experts by merging experts in mixture-of-experts. *arXiv preprint arXiv:2405.11530*, 2024.

[25] Lujun Li, Qiyuan Zhu, Jiacheng Wang, Xiaoyu Qin, Wei Li, Hao Gu, Sirui Han, and Yike Guo. Sub-moe: Efficient mixture-of-expert llms compression via subspace expert merging. In *AAAI*, 2026.

[26] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *ACL*, 2024.

[27] Yushi Huang, Zining Wang, Zhihang Yuan, Yifu Ding, Ruihao Gong, Jinyang Guo, Xianglong Liu, and Jun Zhang. Modes: Accelerating mixture-of-experts multimodal large language models via dynamic expert skipping. *arXiv preprint arXiv:2511.15690*, 2025.

[28] Maryam Akhavan Aghdam, Hongpeng Jin, and Yanzhao Wu. Da-moe: Towards dynamic expert allocation for mixture-of-experts models. *arXiv preprint arXiv:2409.06669*, 2024.

[29] Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *ACL*, 2025.

[30] Zheyue Tan, Zhiyuan Li, Tao Yuan, Dong Zhou, Weilin Liu, Yueqing Zhuang, Yadong Li, Guowei Niu, Cheng Qin, Zhuyu Yao, et al. Rexmoe: Reusing experts with minimal overhead in mixture-of-experts. *arXiv preprint arXiv:2510.17483*, 2025.

[31] Costin-Andrei Oncescu, Qingyang Wu, Wai Tong Chung, Robert Wu, Bryan Gopal, Junxiong Wang, Tri Dao, and Ben Athiwaratkun. Opportunistic expert activation: Batch-aware expert routing for faster decode without retraining. *arXiv preprint arXiv:2511.02237*, 2025.

[32] Shuibai Zhang, Caspian Zhuang, Chihan Cui, Zhihan Yang, Fred Zhangzhi Peng, Yanxin Zhang, Haoyue Bai, Zack Jia, Yang Zhou, Guanhua Chen, et al. Expert-choice routing enables adaptive computation in diffusion language models. *arXiv preprint arXiv:2604.01622*, 2026.

[33] Linye Wei, Zixiang Luo, Pingzhi Tang, and Meng Li. Team: Temporal-spatial consistency guided expert activation for moe diffusion language model acceleration. *arXiv preprint arXiv:2602.08404*, 2026.

[34] Hao Mark Chen, Zhiwen Mo, Royson Lee, Qianzhou Wang, Da Li, Shell Xu Hu, Wayne Luk, Timothy Hospedales, and Hongxiang Fan. Dynamic expert sharing: Decoupling memory from parallelism in mixture-of-experts diffusion llms. *arXiv preprint arXiv:2602.00879*, 2026.

[35] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2023.

[36] 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.

[37] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[38] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[39] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL*, 2022.

[40] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *NAACL*, 2021.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[43] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022.

[44] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

[45] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

[46] Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *ICML*, 2024.

[47] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[48] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[50] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

[51] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024.

[52] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

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

[54] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *International Conference on Learning Representations*, 2026.

[55] Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *Advances in neural information processing systems*, 2025.

[56] Qiuhong Shen, Xingyi Yang, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Speed: Sharpened-teacher distillation for parallel decoding of diffusion language models. In *International Conference on Machine Learning*, 2026.

[57] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

[58] Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.

[59] Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

[60] Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient rl of diffusion large language models. *arXiv preprint arXiv:2510.11683*, 2025.

[61] Sicheng Feng, Zigeng Chen, Xinyin Ma, Gongfan Fang, and Xinchao Wang. dvoting: Fast voting for dllms. *arXiv preprint arXiv:2602.12153*, 2026.

[62] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *Transactions on Machine Learning Research*, 2025.

[63] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[64] Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*, 2025.

[65] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

[66] Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

[67] Fred Zhangzhi Peng, Shuibai Zhang, and Alex Tong. contributors. open-dllm: Open diffusion large language models.

[68] Runpeng Yu, Qi Li, and Xinchao Wang. Discrete diffusion in large language and multimodal models: A survey. *Transactions on Machine Learning Research*, 2026.

[69] Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025.

[70] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. Dmax: Aggressive parallel decoding for dllms. *Advances in neural information processing systems*, 2026.

[71] Rishov Sarkar, Hanxue Liang, Zhiwen Fan, Zhangyang Wang, and Cong Hao. Edge-moe: Memory-efficient multi-task vision transformer architecture with task-level sparsity via mixture-of-experts. In *ICCAD*, 2023.

[72] Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2022.

[73] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021.

[74] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

[75] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

[76] Sicheng Feng, Song Wang, Shuyi Ouyang, Lingdong Kong, Zikai Song, Jianke Zhu, Huan Wang, and Xinchao Wang. Reasonmap: Towards fine-grained visual reasoning from transit maps. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, 2026.

[77] Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. When tokens talk too much: A survey of multimodal long-context token compression across images, videos, and audios. *arXiv preprint arXiv:2507.20198*, 2025.

[78] Junhan Zhu, Hesong Wang, Mingluo Su, Zefang Wang, and Huan Wang. Obs-diff: Accurate pruning for diffusion models in one-shot. *arXiv preprint arXiv:2510.06751*, 2025.

[79] Sicheng Feng, Kaiwen Tuo, Song Wang, Lingdong Kong, Jianke Zhu, and Huan Wang. Rewardmap: Tackling sparse rewards in fine-grained visual reasoning via multi-stage reinforcement learning. In *ICLR*, 2026.

[80] Keda Tao, Kele Shao, Bohan Yu, Weiqiang Wang, Huan Wang, et al. Omnizip: Audio-guided dynamic token compression for fast omnimodal large language models. *arXiv preprint arXiv:2511.14582*, 2025.

[81] Xin Jin, Siyuan Li, Siyong Jian, Kai Yu, and Huan Wang. Mergemix: A unified augmentation paradigm for visual and multi-modal understanding. *arXiv preprint arXiv:2510.23479*, 2025.

[82] Zhenxin Ai and Haiyun He. Pasa: A principled embedding-space watermarking approach for llm-generated text under semantic-invariant attacks. *arXiv preprint arXiv:2605.10977*, 2026.

[83] Wenjie Du, Li Jiang, Keda Tao, Xue Liu, and Huan Wang. Which heads matter for reasoning? rl-guided kv cache compression. In *ICML*, 2026.

[84] Hesong Wang, Xin Jin, Lu Lu, Chenhaowen Li, Jian Chen, Qiang Liu, and Huan Wang. Earlytom: Early token compression completes fast video understanding. In *CVPR*, 2026.

Appendix

In Appendix A, we provide additional preliminaries. In Appendix B, we discuss limitations and future work. We further provide the impact statement in Appendix C and license statement in Appendix D, and computing resources in Appendix E.

A Additional Preliminaries **16**

A.1 Unique Expert Count Across Denoising Steps 16

B Limitations & Future Work **16**

C Impact Statement **17**

D License Statement **17**

E Computing Resources **17**

A Additional Preliminaries

A.1 Unique Expert Count Across Denoising Steps

For Figure 3 (b), following the setting in Observation B, we record the unique expert count at each denoising step and plot its frequency distribution as a histogram. The unique expert count ranges from approximately 45 to 95, indicating nearly twofold variation and substantial fluctuation across denoising steps. We further present a representative sequence of consecutive denoising steps in Figure A1, where the unique expert count exhibits large periodic fluctuations, further supporting Observation B.

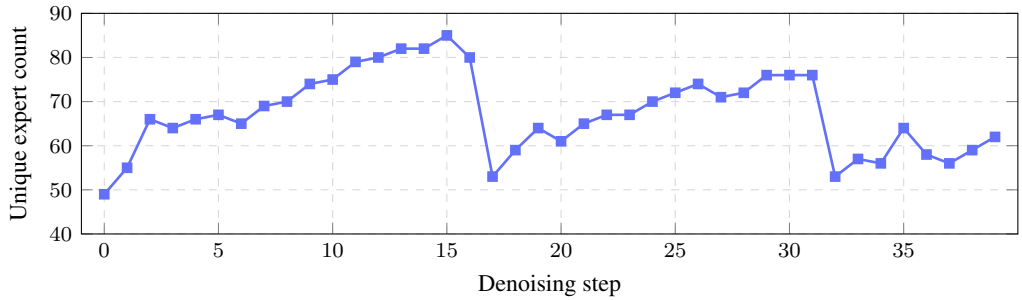


Figure A1: Unique expert count at each step over a representative sequence of consecutive denoising steps.

B Limitations & Future Work

Our method is, in principle, applicable to any MoE dLLMs. In this work, we focus our evaluation on the language modality, but the same idea can be naturally extended to other modalities, such as image and video, and to downstream tasks including visual question answering [76, 77, 78] and visual reasoning [79, 80, 81, 82, 83, 84]. More broadly, our results suggest that block-level expert routing is a promising direction for future MoE dLLMs. Looking ahead, this direction can be pushed further toward more extreme compression, for example, by encouraging all tokens within a block to share the same expert group or by jointly reducing computation cost through selecting fewer experts for each token.

C Impact Statement

This work studies efficient inference for MoE dLLMs by reducing the number of uniquely activated experts during decoding. By implementing block-level expert routing, the proposed method reduces memory overhead, lowers inference cost, and alleviates the memory bottleneck in MoE dLLMs. These improvements make large-scale model deployment more practical, especially in resource-constrained, memory-sensitive, or latency-sensitive settings. More broadly, this work contributes to improving the efficiency and accessibility of dLLMs in real-world applications. Furthermore, the method does not introduce additional ethical or societal concerns.

D License Statement

The model and datasets used in this paper are publicly available, and all experiments are conducted in compliance with their respective licenses. The specific licenses for the model and datasets are listed below.

- LLaDA2.0-mini[5] ..... Apache 2.0 License
- GSM8K[6] ..... MIT License
- MATH500[7] ..... MIT License
- ARC-C[8] ..... cc-by-sa-4.0 License
- MMLU[9] ..... MIT License
- TruthfulQA[10] ..... Apache 2.0 License
- QASPER[11] ..... cc-by-4.0 License

To ensure fair and reproducible evaluation, we follow the official documentation and recommended practices of each model when implementing the inference procedures.

E Computing Resources

The experiments were performed on a server equipped with two AMD EPYC 9654 96-Core processors, 1.5 TiB of system memory, and four NVIDIA H100 GPUs with 80 GB VRAM each. We employed data parallelism to speed up evaluation. Each evaluation run was completed within 12 hours, while each training run finished within 144 hours.

[5] https://huggingface.co/inclusionAI/LLaDA2.0-mini.
[6] https://huggingface.co/datasets/openai/gsm8k.
[7] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[8] https://huggingface.co/datasets/allenai/ai2_arc
[9] https://huggingface.co/datasets/cais/mmlu
[10] https://huggingface.co/datasets/truthfulqa/truthful_qa
[11] https://huggingface.co/datasets/allenai/qasper

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state our contributions and scope, and these claims are supported by the experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work and future directions in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All information needed to reproduce the main results, including training data construction, training details, baselines, and inference and evaluation settings, is provided in Section 5.1. Our code is also publicly available at https://github.com/fscdc/dMoE.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is publicly available at https://github.com/fscdc/dMoE, with instructions for environment setup, training, and evaluation. The training data are constructed from public datasets via self-distillation, as described in Section 5.1.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The training and test details, including data construction, hyperparameters, and inference and evaluation settings, are provided in Section 5.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Sufficient information about experiments has been discussed in Section 5.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware, memory, and running time of our experiments are reported in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This research follows the NeurIPS Code of Ethics properly.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impacts of our work in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets used in this paper are properly cited, and their licenses are listed in Appendix D.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our code and model at `https://github.com/fscdc/dMoE`, together with documentation on environment setup, usage, training, and evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.