TIDE: EFFICIENT AND LOSSLESS MOE DIFFUSION LLM INFERENCE WITH I/O-AWARE EXPERT OFFLOAD

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Large Language Models (dLLMs) have emerged as a competitive alternative to autoregressive (AR) models, offering better hardware utilization and bidirectional context through parallel block-level decoding. However, as dLLMs continue to scale up with mixture-of-experts (MoE) architectures, their deployment on resource-constrained devices remains an open challenge. Existing AR-based methods often incur either prohibitive I/O overhead or significant compute bottlenecks. In this work, we propose ***TIDE***, a novel resource-efficient inference system that leverages the temporal stability of expert activations during the diffusion process within the block. Specifically, we leverage the temporal stability of expert activations during the diffusion process within the block and introduce an interval-based expert refresh strategy that updates the expert placement in an I/O-aware fashion. To ensure optimal performance, we formulate the inference scheduling as a mathematical programming problem, solving for the optimal interval that minimizes I/O traffic and CPU computation. Most importantly, *TIDE* is a lossless optimization that requires no model training, providing a "free lunch" acceleration for dLLM inference. In a single GPU-CPU system, we demonstrate that *TIDE* achieves up to 1.4× and 1.5× throughput improvements over prior baselines on `LLaDA2.0-mini` and `LLaDA2.0-flash` models, respectively.

1 INTRODUCTION

Diffusion-based Large Language Models (dLLMs) have recently emerged as a competitive alternative to autoregressive (AR) Large Language Models (LLMs) (Zhang et al., 2022; Radford et al., 2019; DeepSeek-AI, 2024; Jiang et al., 2024) for text generation tasks. Instead of producing tokens one-by-one in a sequentially left-to-right fashion, dLLMs iteratively denoise multiple masked tokens at the granularity of a block, offering two structural advantages over AR models: (1) each token prediction is conditioned on bidirectional context, allowing for better semantic understanding, and (2) multiple tokens within a block can be decoded in parallel to improve computational efficiency. Built upon this paradigm, a series of open-sourced dLLMs (Nie et al., 2025; Ye et al., 2025; Bie et al., 2025; Wu et al., 2025; Gong et al., 2024) has emerged, most notably the LLaDA series (Nie et al., 2025; Bie et al., 2025), which has achieved performance comparable to its AR counterparts while offering much higher decode throughput (Gong et al., 2024). Most recently, LLaDA-2 (Bie et al., 2025) adopts a sparse mixture-of-experts (MoE) backbone (Fedus et al., 2021; DeepSeek-AI, 2024; Jiang et al., 2024) as AR-based models, in which tokens are routed to a small subset of experts at each layer. This design scales diffusion language models from the original 8B parameters to 100B, making them more production-ready (TWIMLAI, 2026; Fan et al., 2025).

With the ever-increasing popularity of edge computing, running AI models in resource-constrained environments has attracted growing attention in both research and practice (Sheng et al., 2023; Liu et al., 2024; Zhao et al., 2024a;b). Such on-device intelligence both speeds up response latency and enhances data privacy and security, making AI more accessible, efficient, and practical in a wide range of daily applications (Apple, 2024; Microsoft, 2024). Thanks to their inherent parallelism, dLLMs emerged as a compelling option for near-user inference (Wu et al., 2025; Zhang et al., 2025). As the compute capability of edge hardware, such as mobile NPUs and CPUs (Apple, 2024; Zhao et al., 2024a), continues to scale up, dLLMs have become a much more natural fit for on-device uses, achieving significantly higher hardware utilization than memory-bound operations characteristic of token-by-token AR decoding.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

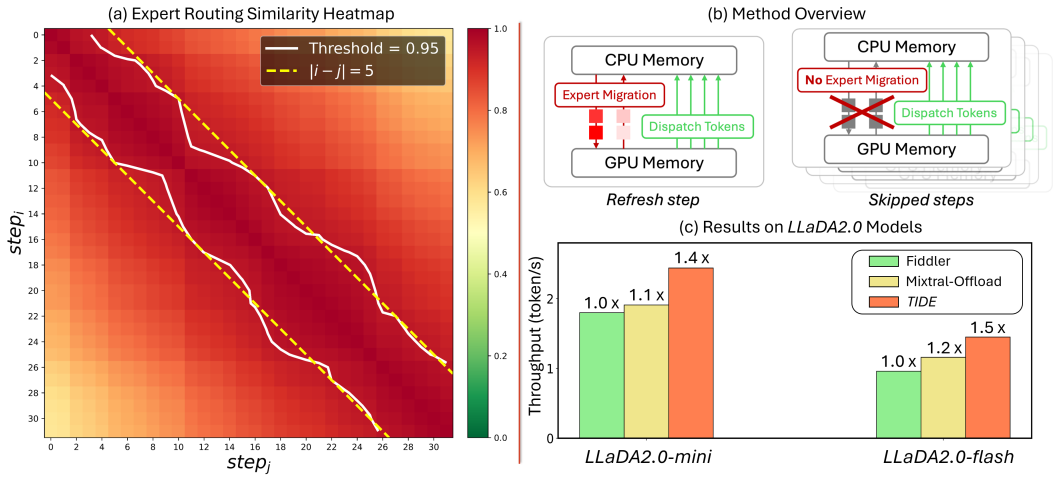

Figure 1: (a) Similarity heatmap of expert routing across denoising steps within a block. Expert routing remains highly similar for nearby steps, and the diagonal bands show that this stability extends beyond immediate neighbors: step pairs separated by five denoising steps retain cosine similarity near 0.95. (b) Overview of *TIDE*. At *refresh steps*, the system intelligently swaps the GPU and CPU experts based on token hit counts (number of tokens each expert has processed). At *skipped steps*, the system continues decoding with the current expert placement and does not migrate experts. By exploiting routing stability across adjacent steps, *TIDE* avoids unnecessary GPU-CPU I/O overhead and maintains high GPU utilization. (c) Throughput comparison of *TIDE* against state-of-the-art MoE inference solutions (Kamahori et al., 2024; Eliseev & Mazur, 2023) for `LLaDA2.0` in a single GPU-CPU setting.

While prior research has achieved promising results on optimizing dense dLLM architectures (typically <8B parameters) (Wu et al., 2025; Ma et al., 2025a; Bao et al., 2025; Li et al., 2025), these methods generally focus on model compression (Shang et al., 2023; He et al., 2023), caching (Wu et al., 2025; Ma et al., 2025a), or efficient decoding (Bao et al., 2025; Li et al., 2025). The efficient deployment of Mixture-of-Experts (MoE) dLLMs (Bie et al., 2025) on resource-limited platforms stands as an open question. Unlike their AR counterparts, MoE-dLLMs present a distinct execution pattern: **In MoE-dLLMs, each denoising step activates experts for all tokens simultaneously within the block. This produces a wide, fragmented expert footprint that could easily trigger out-of-memory (OOM) errors.**

A straightforward solution is to swap experts between GPU and CPU memory (Eliseev & Mazur, 2023; Xue et al., 2024). However, expert migration at every denoising step is prohibitively expensive, as a single dLLM step activates a larger, more diverse set of experts than an AR step, thus creating massive CPU-GPU I/O traffic. An alternative approach is to simply reroute token computation to the CPU experts (Kamahori et al., 2024). But modern CPU execution is often orders of magnitude slower than GPU execution, especially for dense general matrix multiplication (GEMM) operations. The system inevitably becomes CPU-bound as more tokens are routed to the host, causing the GPU to idle while waiting for CPU-processed activations. Consequently, there is an urgent need for an orchestration strategy that achieves (1) *minimal I/O overhead* and (2) *maximal compute efficiency* in the case of inference on resource-constrained systems.

In this work, we propose ***TIDE***, an I/O-aware MoE-dLLM inference system that intelligently schedules the expert routing decisions to improve system throughput with no accuracy drop. **Our key insight is that the expert activation exhibits similar patterns in multiple adjacent denoising steps within a block, thereby creating the opportunity for expert reuse, as shown in Figure 1 (a).** *TIDE* adopts an interval-based expert refresh and reuses the GPU expert set within the same interval. *TIDE* aims to maintain a high GPU expert hit rate while reducing expert migration overhead, which is especially costly in dLLMs because each denoising step routes an entire active block rather than a single new token. Moreover, our method does not require any model training and has no impact on model accuracy, thus offering a free-lunch type acceleration for MoE-dLLM inference.

As shown in Figure 1 (b), *TIDE* splits the decode phase into *refresh steps* and *skip steps*: At refresh steps, *TIDE* promotes the CPU experts with the most token hits to the GPU memory up to its budget. For skipped steps, the model reuses the current placement and routes the tokens to their corresponding expert sets. The optimal interval is determined by modeling the latency overheads using an analytical model and solving a constrained mathematical programming (MP) problem with a combination of hardware profiling and greedy search. Evaluations on both `LLaDA2.0-mini` and `LLaDA2.0-flash` on NVIDIA A100 and H100 GPUs demonstrate that *TIDE* obtains up to 1.4× and 1.5× speedup under different memory constraints against prior works.

In summary, we make the following contributions :

- We identify the challenges for MoE-dLLM inference and propose a new training-free and lossless solution, *TIDE*, for efficient inference on resource-constrained environments.
- By exploiting the cross-step similarity in expert routing, *TIDE* features an interval-based expert refresh strategy that intelligently schedules the expert placement to avoid unnecessary I/O overhead. We optimize the interval choice by formulating and solving the MoE inference as a constrained mathematical programming problem with an analytical model.
- We implement and evaluate *TIDE* on `LLaDA2.0` models in a single GPU-CPU system. Experiments demonstrate that *TIDE* can significantly improve system efficiency over previous baselines without any accuracy drop.

2 Related Work

Diffusion Large Language Models (dLLMs). Diffusion models are a class of generative models that learn to transform noise into data through an iterative denoising process (Ho et al., 2020; Rombach et al., 2022; Peebles & Xie, 2023). They have been widely adopted in image and video generation, where models start from random noise and progressively refine it into high-quality images or videos that align with a given prompt (Chen et al., 2023; Zheng et al., 2024b). Recently, combining diffusion models with LLMs has become a promising direction (Nie et al., 2025; Ye et al., 2025; Bie et al., 2025). Instead of predicting the very next word, they take a block of random noise—or a sequence of masked tokens—and gradually refine it into coherent text (Nie et al., 2025; Ye et al., 2025). This decoding structure provides dLLMs with bidirectional context and enables block-level parallelism during generation. Recent work shows that such a diffusion-based paradigm can scale to a mixture-of-experts (MoE) architecture, with better improved compute efficiency (Bie et al., 2025). In this work, we focus on improving the inference efficiency for MoE-based dLLMs.

Inference Optimization for dLLMs. Due to the rising popularity of dLLMs in both academia and industry, there have been several works focusing on improving their inference-time efficiency, especially for dense models (Ye et al., 2025; Nie et al., 2025). A significant body of work focuses on model compression (Shang et al., 2023; He et al., 2023), improved caching (Wu et al., 2025; Ma et al., 2025a), or efficient decoding strategies (Bao et al., 2025; Li et al., 2025; Israel et al., 2025). Notably, following the KV cache mechanism of AR models (Sheng et al., 2023; Zhao et al., 2024b; Kwon et al., 2023), Fast-dLLM proposes similar block-wise approximate KV caching and a confidence-aware parallel decoding with minimal quality drop (Wu et al., 2025). dKV-Cache exploits the stable KV states in neighboring states to reduce repeated attention computation (Ma et al., 2025a). Learn2PD further introduces a learning-based filter model to avoid redundant decoding and achieve better inference efficiency (Bao et al., 2025). However, from the best of our limited knowledge, there has been no prior work on improving the runtime efficiency of MoE-dLLMs.

Mixture-of-Expert (MoE). MoE-based models have shown promising performance in a wide range of applications and have become the de facto model choice for real-world production systems (DeepSeek-AI, 2024; Jiang et al., 2024; Rajbhandari et al., 2022). Unlike dense models, MoE architectures increase parameter capacity by increasing the number of FFNs (experts), with a subset of experts activated per token to reduce effective computation relative to the total model size (Fedus et al., 2021; DeepSeek-AI, 2024; Jiang et al., 2024). However, deploying MoE models efficiently is particularly challenging due to their large memory footprint, particularly for resource-constrained scenarios. Modern GPU memory cannot hold all the expert weights, creating additional latency overhead of frequent expert swapping between GPU HBM and CPU host memory (Eliseev & Mazur,

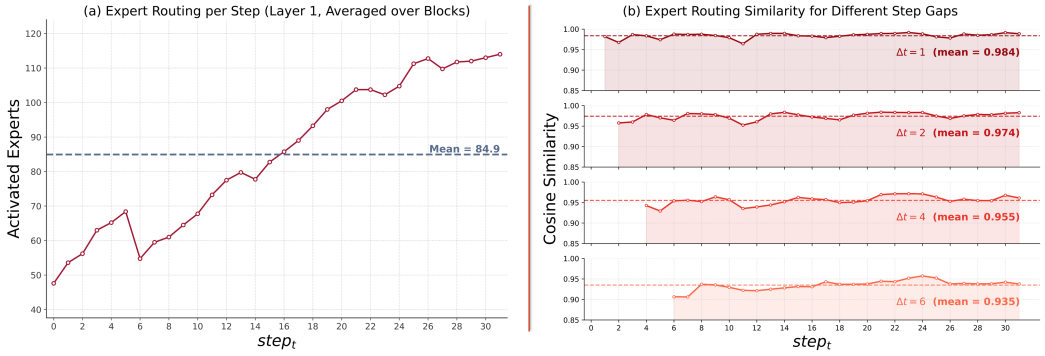

Figure 2: Expert activation pattern with a block of size 32 for `LLaDA2.0-mini`. (a) Number of unique experts activated at each step, which increases as decoding continues. (b) Similarity scores of expert routing for different step intervals at each step within a block. Here, we use the cosine similarity score, following prior works Wu et al. (2025); Bao et al. (2025).

2023; Xue et al., 2024) or slow CPU-based computation (Kamahori et al., 2024). To make matters worse, parallel decoding in MoE-dLLMs activates more experts per step, making prior solutions ill-suited for diffusion-based models.

3 Methodology

In this section, we begin by introducing some preliminary details on the mixture-of-experts (MoE) and formulating the efficiency problem for resource-constrained inference. Next, we present our observations for the expert activation pattern and key insights. Finally, we elaborate on our scheduling and execution strategy for MoE-dLLM, which includes (1) a mathematical programming (MP) model to determine the optimal interval and (2) a detailed description of the expert placement procedure for MoE-dLLM inference.

3.1 Problem Definition

Consider a MoE model (DeepSeek-AI, 2024; Jiang et al., 2024) with L sparse feed-forward network (FFN) layers with E total experts, and k experts are activated for one token at a time. For a batch of tokens, assume K GPU experts are used for token computation, where $K >> k$, the latency of processing tokens in one FFN layer can be formulated as the total of GPU computation time:

$$\mathbf{Lat}^{\text{FFN}} = \mathbf{Lat}^{\text{GPU}}(K) \tag{1}$$

In resource-constrained platforms, GPU memory cannot hold all the expert weights for large MoE models. For instance, Mixtral-8x7B consists of over 46B parameters, requiring over 94 GB of GPU VRAM in FP16, exceeding a single H100 80 GB GPU (Jiang et al., 2024). Here, we assume the GPU holds B number of experts, and the remaining $(E - B)$ experts reside in host memory. The expert selections can be divided as $K = \{K^{\text{GPU}}, K^{\text{CPU}}\}$, and prior methods employ two strategies: (1) reroute tokens to host memory experts for CPU computation or (2) swap the experts between the GPU and the host memory. For token routing, the latency per FFN layer is as follows:

$$\mathbf{Lat}^{\text{FFN}} = \max(\mathbf{Lat}^{\text{GPU}}(K^{\text{GPU}}), \mathbf{Lat}^{\text{CPU}}(K^{\text{CPU}})) \tag{2}$$

And in the case of expert swapping, the latency can be formulated as the GPU computation time and additional expert I/O transfer latency:

$$\mathbf{Lat}^{\text{FFN}} = \begin{cases} \mathbf{Lat}^{\text{GPU}}(K) + \mathbf{Lat}^{\text{I/O}}(K^{\text{CPU}}) & \text{if } K < B \\ \max(\mathbf{Lat}^{\text{GPU}}(K^{\text{GPU}}), \mathbf{Lat}^{\text{CPU}}(K - B)) + \mathbf{Lat}^{\text{I/O}}(B - K^{\text{GPU}}) & \text{if } K > B \end{cases} \tag{3}$$

We can see that latency is highly dependent on the number of existing GPU experts used for token computation, i.e., the *GPU expert hit rate*. For single-batch AR decoding, the number of activated

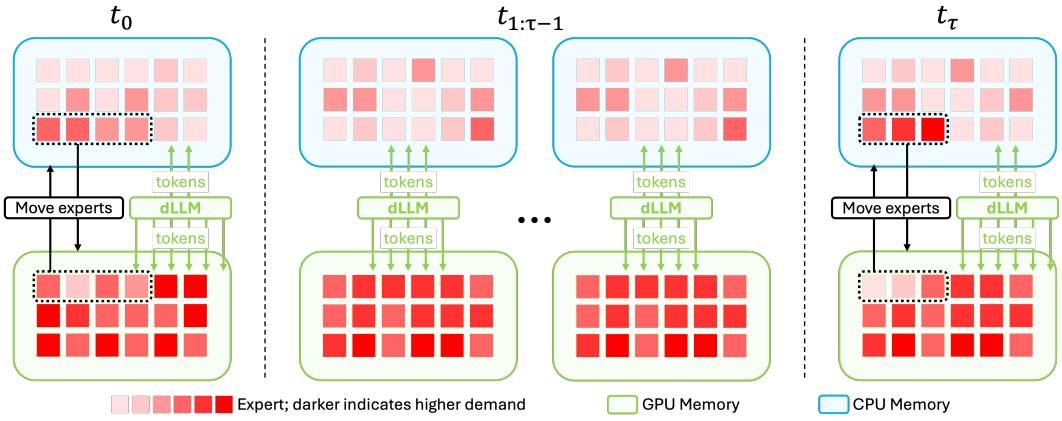

Figure 3: Design of *TIDE*. At **refresh steps** (t_0, t_τ), the system updates the GPU-resident expert set by promoting experts with the highest token hits from CPU memory to GPU memory. Experts outside this set are kept in CPU memory or evicted there if currently GPU-resident. At **skipped steps** $(t_{1:\tau-1})$, decoding continues with the current expert placement and performs no expert migration.

experts stays fixed at $K = k$, which presents not much of an obstacle. However, as shown in Figure 2 (a), in the case of diffusion-based MoE, experts for all tokens within a block are activated, leading to potentially high K. According to equation 3, when the experts on the GPU are not selected for FFN computation $(K > B)$, the inference runtime is bottlenecked by both the CPU computation and GPU-CPU I/O, creating potentially severe inference bottlenecks. **Given this efficiency obstacle in MoE inference, we need to find a scheduling policy that orchestrates both the expert migration and token routing in resource-constrained systems, so that the overall execution time is minimized**.

3.2 Observation & Insights

Since diffusion-based generation is inherently an iterative reverse process, each token is progressively denoised in a coarse-to-fine manner, evolving from `[MASK]` toward a concrete `[word]` prediction (Ho et al., 2020; Rombach et al., 2022; Nie et al., 2025). As a result, the latent representations produced at adjacent denoising steps often exhibit strong similarity, a property that has been observed in prior work (Ma et al., 2025a; Wu et al., 2025). Motivated by this observation, we investigate whether a similar form of cross-step stability also arises in expert activation during MoE-dLLM inference.

Figure 2 (b) shows that adjacent denoising steps indeed induce highly similar expert activation patterns. We highlight two key observations. First, expert activation exhibits strong temporal locality. The set of activated experts changes only marginally between consecutive denoising steps, with a mean within-block cosine similarity of 0.985. This finding aligns with prior observations that intermediate features in diffusion models can be effectively reused across nearby denoising steps (Ma et al., 2025a; Wu et al., 2025; Bao et al., 2025). Second, routing similarity remains high not only between immediate neighbors but also within a broader band around the diagonal. In particular, step pairs separated by as many as five denoising iterations still retain a cosine similarity above 0.95.

The above observations indicate that routing decisions at one step are highly predictive of expert demand in subsequent steps, suggesting that the expert activation distribution can be treated as approximately quasi-static over short denoising intervals. This temporal stability has an important practical implication: rather than recomputing or adapting expert-related decisions independently at every denoising step, MoE-dLLM inference can potentially amortize such decisions across a short window of steps. There, it creates an opportunity to exploit routing locality for more efficient inference while preserving the model's dynamic expert selection behavior.

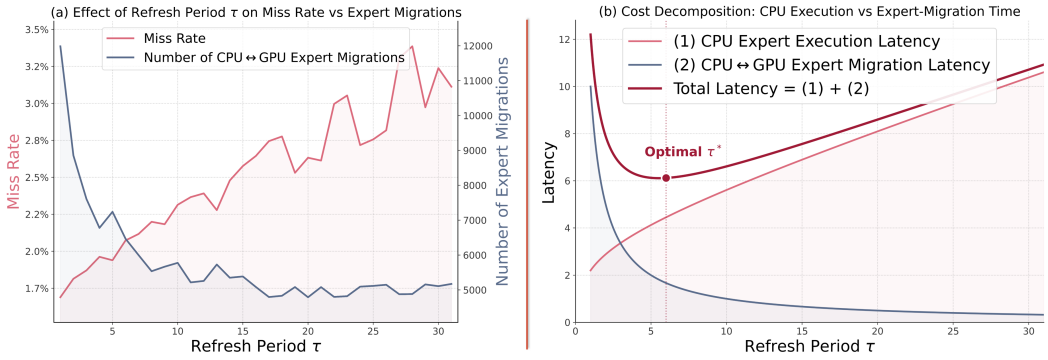

Figure 4: Impact of the refresh interval τ. (a) Relationship of GPU expert miss rate and the number of expert migrations with respect to τ. Increasing τ generally raises the GPU expert *miss rate*. Meanwhile, a larger τ reduces the number of expert migrations. The migration curve shows diminishing returns at larger τ, consistent with our drift analysis in Eq. 5. (b) Relationship of expert migration and CPU computation latency with respect to τ based on our analytical model. We can see that an optimal τ can be determined by solving the optimization problem in Eq. 7.

3.3 *TIDE* DESIGN

Given the above-mentioned findings, we propose *TIDE*, which leverages the temporal locality of expert activation patterns to intelligently make the expert swapping and token routing decisions in a training-free manner. Specifically, *TIDE* introduces an expert refresh strategy that swaps the experts between GPU memory and host memory at the interval of τ steps ($\tau > 1$) within a block. As shown in Figure 3, *TIDE* partitions the decoding process within a block into two distinct phases: *refresh steps* and *skipped steps*. At refresh steps (e.g., t_0 or t_τ), *TIDE* dynamically updates the GPU-resident expert set by promoting 'high-demand' experts on the host with the highest token hits to GPU memory, while evicting 'low-demand' experts back to CPU memory. During the intervening skipped steps (t_1 to $t_{\tau-1}$), a fixed expert placement is maintained with no migration. This hybrid approach ensures that the majority of computation remains on the GPU, significantly amortizing the overhead of expert swapping and CPU computation. Since *TIDE* only changes to perform load balancing between GPU and CPU experts, it has no impact on model outputs.

Interval-based Expert Refresh. In the design of *TIDE*, a key question to answer is how to decide the optimal interval τ. As shown in equations 2 and 3, the dominant costs of offloaded MoE inference are expert migration (I/O) and CPU expert executions. To better understand the tradeoffs between I/O and CPU computation, we define the mismatch between the expert set in step $t-1$ and t, as *drift rate* as:

$$d_t = \frac{|\Delta K_t^{\text{GPU}}|}{B} \tag{4}$$

where ΔK_t^{GPU} denotes the expert selection difference between the current optimal placement and the previous optimal placement. Assuming independent per-expert replacement events, the probability that any given expert is still optimal after τ steps is $\prod_{t=1}^{\tau-1}(1-d_t)$, so the expected number of experts that need to be migrated at the next refresh is $B \cdot (1 - \prod_{t=1}^{\tau-1}(1-d_t))$. From Figure 1(a), we see that cross-step similarity scores exhibit high consistency, d_t can be approximated as a constant d. Over T denoising steps, the total expected migration latency costs between GPU and CPU can be treated as a function of τ:

$$\mathbf{Lat}^{\text{I/O}}(\tau) \approx C^{\text{I/O}} \cdot \frac{B \cdot T}{\tau} \cdot \big(1 - (1-d)^\tau\big). \tag{5}$$

where $C^{\text{I/O}}$ is a GPU-CPU I/O bandwidth-related constant. At $\tau = 1$ this recovers the full-refresh baseline $M(1) = B \cdot T \cdot d$. As τ grows, $1-(1-d)^\tau \to 1$ and $M(\tau) \to BT/\tau$, exhibiting the $1/\tau$ scaling and diminishing returns visible in Figure 4 (a).

However, increasing the refresh interval τ comes at a cost. Although adjacent steps have highly similar routing, the similarity drops when steps get farther away, as shown in Figure 1 (a) and

Algorithm 1 *TIDE* Scheduling Policy for dLLM-MoE Inference

Require: Full expert set $\mathcal{E}$ with CPU expert set $\mathcal{E}^{\text{CPU}}$ and GPU expert set $\mathcal{E}^{\text{GPU}}$, the number of GPU experts B, Hit counter $\mathcal{H}$, refresh interval τ, block size T and decoding step t, token states $X = \{(x, \mathcal{E}_x)\}$ has its token x and corresponding expert selections $\mathcal{E}_x$.

for all $t < T$ **do**
  if $t \,\%\, \tau == 0$ **then** ▷ Update Expert Placement at τ intervals
    $\mathcal{E}^{\text{GPU}}_{B,t} = \text{TopB}_{e \in \mathcal{E}}\, \mathcal{H}_t(e)$ ▷ Get the top B experts with highest token hit count
    $\mathcal{E}^{\uparrow}_t = \mathcal{E}^{\text{GPU}}_{B,t} \setminus \mathcal{E}^{\text{GPU}}_{B,t-1}$ ▷ Experts to upload to GPU
    $\mathcal{E}^{\downarrow}_t = \mathcal{E}^{\text{GPU}}_{B,t-1} \setminus \mathcal{E}^{\text{GPU}}_{B,t}$ ▷ Experts to be evicted to CPU
    $\mathcal{E}^{\downarrow}_t \xrightarrow{\text{PCIe}} \mathcal{E}^{\text{CPU}}, \mathcal{E}^{\uparrow}_t \xrightarrow{\text{PCIe}} \mathcal{E}^{\text{GPU}}$ ▷ Migrate experts
  # Token Routing
  for all $(x, \mathcal{E}_x)$ in X **do** ▷ Token Routing to Their Respective Experts
    for all e in $\mathcal{E}_x$ **do**
      if e in $\mathcal{E}^{\text{CPU}}$ **then**
        $x \xrightarrow{\text{PCIe}} \mathcal{E}^{\text{CPU}}$ ▷ Route tokens to CPU experts
      output $= e(x)$ ▷ Process Tokens on both GPU and CPU

Figure 2 (b). At a refresh step, K^{GPU}_t is set to the current top-B experts to maximize GPU expert hit rate. During skipped steps, the token hit rate continues to fall, as shown in Figure 4 (a), leading to increased CPU computation time. The expected CPU computation costs can be defined as:

$$\mathbf{Lat}^{\text{CPU}}(\tau) \approx C^{\text{CPU}} \cdot T \cdot B \cdot f(\tau). \tag{6}$$

where C^{CPU} is a CPU-related computation constant and $f(\cdot)$ is a general monotonically increasing function for τ, i.e., $\tau_1 \leq \tau_2 \implies f(\tau_1) \leq f(\tau_2)$. To find the optimal τ, we need to minimize the total costs, $\mathbf{Lat}^{\text{total}}(\tau) = \mathbf{Lat}^{\text{CPU}}(\tau) + \mathbf{Lat}^{\text{I/O}}(\tau)$, and solve the following mathematical programming (MP) problem:

$$\min_{\tau \in [1,2,\ldots,T-1]} \left(\frac{B \cdot T}{\tau} \cdot \left(1 - (1-d)^{\tau}\right) + C^{\text{CPU}} \cdot T \cdot B \cdot f(\tau)\right) \tag{7}$$

We solve this problem by first running a hardware profiling on the CPU computation speed and I/O bandwidth performance to approximate constants C with different prompt and output configurations to create a mapping between input configurations and their execution time. Next, we apply a greedy search method to solve the optimization problem for the best performance. This process is done offline, introducing no overhead during the actual inference process.

Expert Selection and Token Routing. Another key question to answer is how to perform appropriate expert swapping, i.e., which experts are offloaded and uploaded. To this end, we employ a global hit counter, where we calculate the expert activation hits for all the experts at refresh steps, select the top experts by hit frequency ranking, and swap the expert sets on GPU and CPU to maximize reuse potential during skipped steps. The details of our scheduling policy during MoE-dLLM inference are shown in Algorithm 1.

Lossless Inference. Since *TIDE* focuses on expert placement without modifying the selection of MoE router or model weights, and does not alter the parallel decoding mechanism, it preserves the model outputs, thus inducing no accuracy degradation.

4 Experiments

4.1 Experimental Settings

Models and Datasets. We evaluate our method on the `LLaDA2.0` architecture (Bie et al., 2025), namely, `LLaDA2.0-mini` (16BA1B) and `LLaDA2.0-flash` (100BA6B). Both models have a total of 256 FFN experts with $top_k = 8$ activation pattern. We use the sanitized MBPP dataset (Austin et al., 2021) from the `lm_eval_harness` library (Gao et al., 2026). Since our method is essentially lossless, it can be easily generalized to other datasets.

Table 1: Performance comparison of our method with Fiddler (Kamahori et al., 2024) and Mixtral-Offload (Eliseev & Mazur, 2023) for `LLaDA2.0` models on the sanitized MBPP benchmark. TPS denotes the number of tokens decoded per second (higher is better). All runs use a block length of 32 and a confidence threshold of 0.95.

	Gen Length	GPU Expert Budget	GPU Memory Constraint	Throughput (token/s) ↑		
				Fiddler	Mixtral-Offload	*TIDE* (Ours)
`LLaDA2.0-mini`	256	64	10 GB	1.81	1.69	**2.11**
		128	18 GB	1.74	1.76	**2.36**
	1024	64	10 GB	1.79	1.45	**1.89**
		128	18 GB	1.80	1.91	**2.44**
`LLaDA2.0-flash`	256	32	30 GB	0.95	1.01	**1.25**
		64	55 GB	1.14	1.35	**1.73**
	1024	32	30 GB	0.89	1.01	**1.24**
		64	55 GB	1.05	1.16	**1.45**

Hardware and Implementations. We run `LLaDA2.0-mini` on an NVIDIA A100 40 GB GPU and `LLaDA2.0-flash` on an NVIDIA H100 80 GB GPU, with a 48-Core Intel CPU and 1024 GB DDR4 host memory. To ensure high portability and reproducibility, we implement *TIDE* on top of HuggingFace Transformers (Wolf et al., 2019) and dInfer (Ma et al., 2025b), with PyTorch 2.9, CUDA 12.8, which can be further incorporated into popular serving frameworks, such as SGLang (Zheng et al., 2024a) and vLLM (Kwon et al., 2023).

Baselines and Metrics. Since there is no prior work on optimizing dLLM-MoE inference, we compare *TIDE* with two prior baseline methods for AR-MoE models. Specifically, we use Fiddler (Kamahori et al., 2024) as a CPU computation baseline, where expert placements remain static during decoding, and Mixtral-Offloading (Eliseev & Mazur, 2023), which performs expert offloading at each denoising step. We compare *TIDE* and prior baselines by evaluating inference efficiency with decode throughput, calculated as the average number of decoded tokens per second (token/s).

4.2 Main Results

Table 1 demonstrates the system efficiency comparison of our methods and prior solutions. Overall, we observe that *TIDE* offers the highest attainable throughput for MoE-dLLM inference in resource-constrained systems. There are three key observations. First, *TIDE* achieves consistent speedup over all baselines, showing 1.2∼1.4× higher throughput over Mixtral-Offload. This is due to the fact that prior works do not consider the unique expert activation pattern for diffusion-based MoE models, thereby leading to suboptimal performance. Second, the speedup of *TIDE* is consistent across different generation lengths. While Fiddler achieves comparable performance under certain settings, its efficacy drops when scaling to the larger 100B model with more intensive computation. Third, *TIDE* shows much better performance, especially when GPU memory capacity is limited. This is due to the intelligence expert migration policy of *TIDE*, which maximizes the expert reuse and avoids redundant I/O transfer overheads.

4.3 Performance Analysis

Impact of the Refresh Interval τ. Here, we showcase the efficacy of our refresh interval optimization method. We compare the throughput performance with different refresh interval configurations, i.e., (1) $\tau = 1$ (Mixtral-Offload (Eliseev & Mazur, 2023)), (2) random choice, and (3) optimized τ.

We conduct experiments on both `LLaDA2.0-mini` and `LLaDA2.0-flash` with varying GPU expert budgets (64/128 for LLaDA2.0-mini, and 32/64 for LLaDA2.0-flash), varying block lengths (32/64) with confidence threshold 0.95, and generation length 1024, with results shown in Table 2. Here, we summarize two insights. First, our method consistently delivers the highest throughput out of the three interval choices. Notably, it provides up to 1.4× speedup against random τ, which validates our approach of formulating and solving the mathematical programming problem. Second, *TIDE* can sustain robust performance improvement with respect to both block sizes and GPU budget

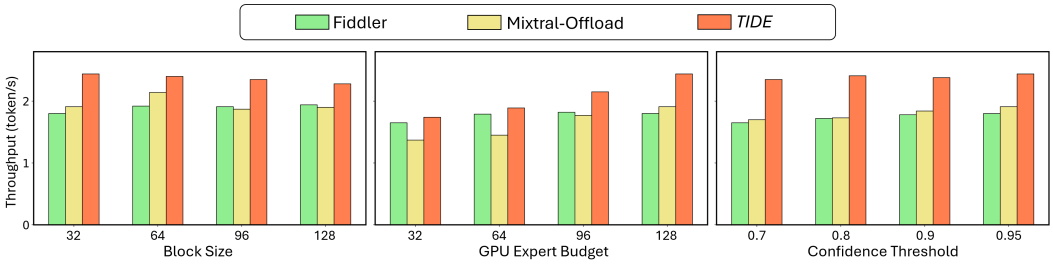

Figure 5: Performance analysis for `LLaDA2.0-mini` on NVIDIA A100 40 GB GPU. From left to right are the throughput comparisons of different methods over varying block sizes (32∼128), GPU expert budgets (32∼128), and confidence thresholds (0.7∼0.95). We can see that *TIDE* consistently outperforms baseline methods regardless of decoding settings.

experts. It can achieve better speedup against prior baseline (Eliseev & Mazur, 2023), especially in the case of higher GPU expert budgets. This is thanks to the interval-based strategy that avoids redundant I/O transfer of expert weights.

Sensitivity Studies. We further analyze the impact of block size, GPU expert budget, and confidence threshold on the end-to-end system throughput of `LLaDA2.0-mini` on NVIDIA A100 40 GB GPU, with results shown in Figure 5. There are three key observations here. First, *TIDE* achieves the highest throughput performance against different block sizes with consistent improvement against baseline methods. This is due to the fact that our interval-based strategy can (1) improve GPU expert hit rate, reduce the tokens routed to CPU, thus maximizing compute efficiency, and (2) minimize I/O expert transfer overhead by using the optimal interval determined by solving an optimization problem. Second, both Mixtral-offload and *TIDE* scales well with GPU expert budgets, i.e., GPU memory constraints, while Fiddler does not. This highlights the importance of expert placement in MoE-dLLM inference, since Fiddler is increasingly bottlenecked by CPU computation. Third, *TIDE* also sustains consistent improvement under different confidence thresholds, with an average speedup of 1.4× over prior baselines.

Table 2: Throughput (tokens/s) on sanitized MBPP for different choices of refresh interval τ, block sizes, and GPU expert budgets.

Block Size	GPU Expert Budget	`LLaDA2.0-mini` Throughput (tokens/s)		
		$\tau = 1$	Random τ	Optimal τ
32	64	1.79	1.62	**1.89**
32	128	1.91	2.14	**2.44**
64	64	1.71	1.65	**1.90**
64	128	2.14	2.24	**2.40**
Block Size	GPU Expert Budget	`LLaDA2.0-flash` Throughput (tokens/s)		
		$\tau = 1$	Random τ	Optimal τ
32	32	1.01	0.85	**1.24**
32	64	1.16	1.23	**1.45**
64	32	1.12	0.95	**1.32**
64	64	1.27	1.32	**1.49**

5 Conclusion

This paper proposes *TIDE*, a resource-efficient and I/O-aware inference system for MoE-based diffusion language models. By exploiting the unique expert activation patterns during decoding, *TIDE* utilizes an interval-based expert refresh strategy to update expert placement periodically, thus avoiding redundant I/O transfer and CPU computation, thereby improving end-to-end system decode throughput in resource-constrained scenarios. Evaluations demonstrate that *TIDE* achieves up to 1.4×, and 1.5× throughput improvement on `LLaDA2.0-mini` and `LLaDA2.0-flash` models, respectively, in a single GPU-CPU system.

AI USE STATEMENT

In this work, we used generative AI tools to assist with converting author-developed mathematical expressions into LaTeX and provide feedback on research methodology or experiments. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design research methodology or experiments, implement methods, assist with translation, support qualitative and thematic data analysis, interpret results, and generating synthetic data sets, cleaning and reformating dataset are not applicable to this work. Additionally, we used generative AI tools to create and modify the code that generates scientific figures or images, draft software code, summarize or analyse existing literature, sourcing/searching for information, edit the paper to improve readability, identify relevant literature, format references, suggest a structure for the paper. We have reviewed all AI-assisted work. We used generative AI tools to proofread and polish author-written text. All AI-assisted text and code were carefully reviewed and revised by the authors to ensure correctness and preserve the intended meaning. For figures, AI assistance was limited to drafting plotting code; no figure was generated directly by an image-generation model. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Experimental settings and implementation details are provided in Section 4. We also provide the source code to support reproduction of our results in the supplementary materials.

REFERENCES

Apple. Apple intelligence, 2024. URL https://www.apple.com/apple-intelligence/.

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

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. *ArXiv*, abs/2509.25188, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b. *ArXiv*, abs/2512.15745, 2025. URL https://arxiv.org/abs/2512.15745.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James T. Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *ArXiv*, abs/2310.00426, 2023.

DeepSeek-AI. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437, 2024.

A. V. Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading. *ArXiv*, abs/2312.17238, 2023. URL https://api.semanticscholar.org/CorpusID:266573098.

Jiakun Fan, Yanglin Zhang, Xiangchen Li, and Dimitrios S. Nikolopoulos. Taming the memory footprint crisis: System design for production diffusion llm serving. *ArXiv*, abs/2512.17077, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021. URL https://arxiv.org/abs/2101.03961.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 01 2026. URL https://zenodo.org/records/18394108. Code available at https://github.com/EleutherAI/lm-evaluation-harness.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. *ArXiv*, abs/2410.17891, 2024.

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *ArXiv*, abs/2305.10657, 2023.

Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

Daniel Mingyi Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *ArXiv*, abs/2506.00413, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *ArXiv*, abs/2401.04088, 2024.

Keisuke Kamahori, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models. *ArXiv*, abs/2402.07033, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. *ArXiv*, abs/2508.19982, 2025.

Zechun Liu, Changsheng Zhao, Forrest N. Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *International Conference on Machine Learning*, 2024.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *ArXiv*, abs/2505.15781, 2025a. URL https://api.semanticscholar.org/CorpusID:278782363.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, Zhenzhong Lan, Junbo Zhao, Jianguo Li, and Da Zheng. dinfer: An efficient inference framework for diffusion language models. *ArXiv*, abs/2510.08666, 2025b.

Microsoft. Introducing copilot+ pcs, 2024. URL https://blogs.microsoft.com/blog/2024/05/20/introducing-copilot-pcs/.

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*, abs/2502.09992, 2025. URL https://arxiv.org/abs/2502.09992.

William S. Peebles and Saining Xie. Scalable diffusion models with transformers. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale, 2022. URL https://arxiv.org/abs/2201.05596.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2022.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1972–1981, 2023.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu, 2023. URL https://arxiv.org/abs/2303.06865.

TWIMLAI. The race to production-grade diffusion llms with stefano ermon, 2026. URL https://twimlai.com/podcast/twimlai/race-production-grade-diffusion-llms.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

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

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh K. Marina. Moe-infinity: Activation-aware expert offloading for efficient moe serving. *ArXiv*, abs/2401.14361, 2024. URL https://api.semanticscholar.org/CorpusID:267211688.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, and et al. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022.

Tianao Zhang, Zhiteng Li, Xianglong Yan, Haotong Qin, Yong Guo, and Yulun Zhang. Quantdllm: Post-training extreme low-bit quantization for diffusion large language models. *ArXiv*, abs/2510.03274, 2025.

Youpeng Zhao, Ming Lin, Huadong Tang, Qiang Wu, and Jun Wang. Merino: Entropy-driven design for generative language models on iot devices. In *AAAI Conference on Artificial Intelligence*, 2024a.

Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *ArXiv*, abs/2403.17312, 2024b.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024a.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *ArXiv*, abs/2412.20404, 2024b.