Dynamic Expert Sharing: Decoupling Memory from Parallelism in Mixture-of-Experts Diffusion LLMs

Hao (Mark) Chen[1], **Zhiwen Mo**[1], **Royson Lee**[2], **Qianzhou Wang**[1], **Da Li**[2], **Shell Xu Hu**[2], **Wayne Luk**[1], **Timothy Hospedales**[2], **Hongxiang Fan**[1]
[1]Imperial College London, London, UK [2]Samsung AI Center, Cambridge, UK

Abstract

Among parallel decoding paradigms, diffusion large language models (dLLMs) have emerged as a promising candidate that balances generation quality and throughput. However, unlike traditional autoregressive (AR) decoding, this paper identifies a critical bottleneck in dLLMs with Mixture-of-Experts (MoE) architectures: "**expert explosion**". As the number of parallel tokens generated in dLLMs increases, the number of distinct experts activated grows nearly linearly. This results in substantial memory traffic that pushes inference into a memory-bound regime, negating the efficiency gains and scalability of both MoE and parallel decoding. To address this challenge, we propose **Dynamic Expert Sharing (DES)**, a novel technique that shifts MoE optimization from conventional token-centric pruning and expert skipping to sequence-level expert sharing. To maximize expert reuse, DES identifies a compact, high-utility set of experts for an entire parallel decoding block and further improves system efficiency with custom GPU kernels. Extensive experiments on MoE dLLMs demonstrate that, on LLaDA2.0-Mini, DES reduces unique expert activations by up to 55% and latency by up to 38%, while retaining up to 99% of vanilla accuracy, effectively decoupling memory overhead from the degree of parallelism. Code is available at `https://github.com/hmarkc/DES`.

1 Introduction

Recent advances in large-scale foundation models have seen the emergence of native parallel decoding paradigms, most notably diffusion large language models (dLLMs) [39, 36]. The state-of-the-art dLLMs typically leverage block-based parallel decoding [3, 34] to achieve performance parity with traditional autoregressive (AR) models through concurrent token generation. To scale these architectures effectively, dLLMs increasingly integrate Mixture-of-Experts (MoE) [15, 35, 1] as a core architectural component [5, 39, 10]. The primary advantage of MoE lies in its ability to decouple the model's overall knowledge capacity from its per-token computational cost. By activating only a sparse subset of experts for each input, MoE allows dLLMs to scale to massive parameter counts, significantly enhancing representational power without a proportional increase in FLOPs.

The integration of MoE into dLLMs, however, introduces an efficiency paradox, illustrated by our roofline analysis (Figure 1b). While transitioning from AR to parallel dLLM increases operational intensity, the sparse activation of MoE dLLMs makes them more memory-bound than their dense counterparts. In these memory-bound scenarios, typical of multi-GPU and CPU-offloading environments [6, 37], latency is dominated by the traffic from High Bandwidth Memory (HBM) to Static RAM (SRAM) required to load unique expert weights. This bottleneck is intensified by an "**expert explosion**" inherent to parallel generation: whereas AR decoding processes a single token per step, dLLMs process N tokens simultaneously (Figure 1a). Consequently, if experts are selected independently, the unique expert load scales nearly linearly with the block size N, rapidly exceeding the steady-state memory traffic of AR decoding (Figure 1b). Crucially, as the parallel block scales, the overhead of transferring divergent experts from HBM to on-chip memory can negate

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

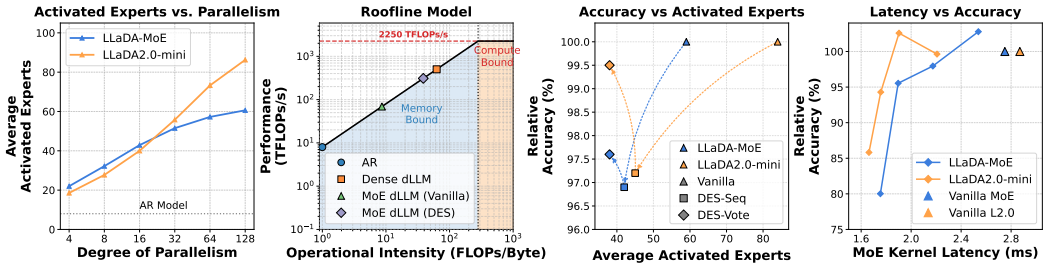


Figure 1: From left to right: (a) Growth of unique activated experts vs. block size. (b) Roofline model showing MoE dLLMs are more memory-bound than dense counterparts and how DES mitigates this bottleneck. (c) Activated experts vs. relative accuracy, where DES-Vote extends the Pareto frontier. (d) MoE kernel latency vs. relative accuracy. Accuracy in (c, d) is relative to the vanilla baseline. **Setup**: single B200, batch size 1, generation length 2048; block size 32 except (a) sweeps 4–128. (a–c) average over HumanEval, MBPP, MATH500, GSM8K; (d) uses MBPP.

the compute savings of MoE, potentially rendering these models slower than their dense equivalents. This overhead imposes a systemic ceiling on the throughput gains promised by parallelization.

Existing dynamic efficiency optimizations for MoE, such as expert skipping [23, 16, 2], are primarily designed for AR models and remain strictly token-centric. While effective at reducing per-token computational FLOPs, these methods prioritize local expert sparsity rather than exploiting cross-token semantic coherence to reduce expert redundancy. Consequently, they fail to mitigate the global HBM traffic bottleneck, as the unique expert weights loaded per layer are not optimized at the sequence level. Crucially, these approaches do not directly address the memory traffic overhead encountered in block-wise parallel decoding, where the collective expert footprint, rather than per-token compute, becomes the primary constraint on throughput.

To address these challenges, we propose **Dynamic Expert Sharing (DES)**, a novel method designed to minimize the expert footprint by forming consensus across parallel tokens. Our approach shifts the optimization objective from per-token pruning to sequence-level sharing. We hypothesize that, unlike standard AR batching where independent tokens may activate disparate experts, *parallel decoding in dLLMs inherently involves semantically coupled tokens that exhibit significant overlap in expert demand*. To exploit this, we introduce **coreset selection**, where a minimal, high-utility set of experts is dynamically identified at runtime to serve the entire parallel block. We explore two specific strategies: **(1) Intra-Sequence Sharing (DES-Seq):** We take the union of experts chosen independently by each token in the block. This allows for vectorized execution, eliminating the need to handle divergent expert assignments for individual positions. **(2) Saliency-Aware Voting (DES-Vote):** A novel mechanism where tokens collectively vote for experts based on their weighted router saliency. This prioritizes the most critical experts by focusing on the holistic semantic complexity of the sequence.

By restricting expert selection to a shared coreset, we significantly reduce HBM-to-SRAM traffic, effectively mitigating the memory traffic overhead of parallel MoE decoding at a small accuracy cost. As shown in Figure 1c and Figure 1d, DES reduces the active experts by up to 55% and the MoE kernel latency by up to 38.0% while achieving performance comparable to the vanilla baseline.

Our contributions are summarized as follows:

1. We identify and quantify the "expert explosion" bottleneck in parallel MoE decoding, demonstrating how independent expert selection across parallel tokens induces a critical memory traffic overhead (Section 3).

2. We propose **Dynamic Expert Sharing**, a method that shifts MoE optimization from token-centric pruning to sequence-level expert sharing, leveraging a novel saliency-aware voting mechanism to maximize expert reuse (Section 4).

3. Through extensive benchmarking on MoE dLLMs, we demonstrate that DES achieves significant memory saving and speedup with negligible accuracy drop, effectively decoupling memory traffic from the degree of parallelism (Section 5).

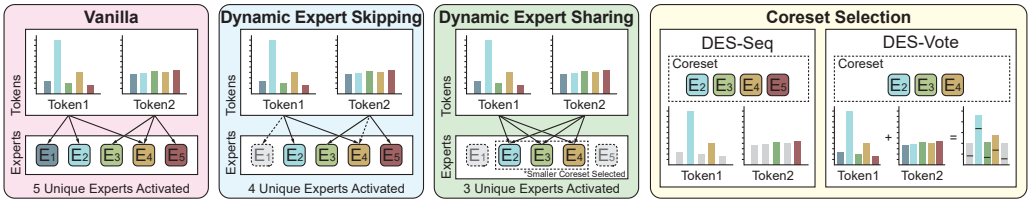


Figure 2: Overview of the Dynamic Expert Sharing method. (a) Vanilla MoE dLLM independently routes multiple tokens, leading to many unique activated experts. (b) Dynamic Expert Skipping reduces local per-token computation (indicated by dotted lines) but often fails to optimize the global unique expert load. (c) Dynamic Expert Sharing employs sequence-level coreset selection (via DES-Seq or DES-Vote) to identify high-utility experts globally, minimizing unique expert weights transferred from HBM. (d) DES-Vote enforces sequence-level consensus by aggregating router weights across the parallel block, contrasting with the independent per-token selection in DES-Seq. Greyed out boxes and bars represent experts not selected and not fetched to on-chip memory.

2 Background

Parallel Decoding. Parallel decoding has been adopted for LLMs to reduce the number of memory-bound forward passes. These approaches utilize either trained tokens [7, 21] or n-grams retrieved from context [13] as inputs to an AR model, enabling the parallel generation of multiple tokens. However, limited backbone re-training caps generation quality, restricting these models to speculative decoding settings that require an extra verification step.

Diffusion Large Language Models. To fully realize the acceleration potential of parallel generation, natively parallel models [14, 39] are being trained from scratch with multi-token prediction objectives. In particular, Diffusion Large Language Models (dLLMs) have emerged as a promising candidate to balance generation quality and efficiency [39, 36, 3]. These models ingest multiple masked tokens and process them concurrently, enabling parallel decoding without the need for verification steps. To further enhance dLLM throughput, recent research has introduced approximate KV cache and sampling strategies [34, 33, 22] alongside specialized inference frameworks [25]. In concurrent work, TEAM [32] accelerates MoE dLLMs by leveraging temporal-spatial consistency with adaptive decoding and speculative exploration strategies.

Efficient Mixture-of-Experts. Mixture-of-Experts (MoE) has emerged as the de facto architecture to scale up dLLMs [5, 39, 10]. Formally, an MoE layer replaces the standard feedforward network (FFN) with a set of M independent experts $\{E_1, \ldots, E_M\}$, where each expert E_i is a sub-network and D is the hidden dimension. A gating or routing function $G : \mathbb{R}^D \to \mathbb{R}^M$ assigns a score to each expert for a given input hidden state x. The output of the MoE module is computed as the weighted sum of the K highest-scoring experts: $\text{MoE}(x) = \sum_{i \in \mathcal{S}} \frac{G(x)_i}{\sum_{j \in \mathcal{S}} G(x)_j} E_i(x)$, where $\mathcal{S} = \text{TopK}(G(x), K)$ is the set of indices corresponding to the K experts with the highest routing scores. For dLLMs, the total number of unique experts required across a batch of N tokens is $|\bigcup_{n=1}^{N} \mathcal{S}_n|$, which typically grows as the degree of parallelism increases. Existing strategies for MoE efficiency primarily follow two paradigms: offline expert pruning and dynamic expert skipping. Offline approaches, such as expert pruning [9, 38] and merging [19], aim to reduce the static parameter footprint through structural sparsity, quantization, or weight consolidation. In contrast, dynamic expert skipping [17, 23, 16, 2] leverages runtime information to bypass redundant expert computations on-the-fly. Additionally, recent works [29, 31, 26] attempted expert sharing across batches for AR decoding.

Key Difference. DES is fundamentally distinct from prior dynamic MoE optimizations along three axes: *(i) sharing vs. skipping*: rather than activating fewer experts per token to save FLOPs, DES concentrates a shared expert pool across tokens to cut unique HBM traffic; *(ii) sequence-level vs. token-level*: DES aggregates routing information across the entire parallel block into a collective decision instead of making isolated per-token ones; and *(iii) sequence vs. batch dimension*: DES exploits the semantic coupling of tokens within a single block, orthogonal to batch-level sharing (e.g., OEA [29]) and to offline pruning. Existing online optimizations are largely tailored for AR models and do not target the memory-traffic bottleneck inherent to dLLM parallel decoding.

3 Motivation

The primary motivation for parallel decoding in dLLMs is to overcome the sequential bottleneck of AR generation, which is notoriously memory-bound due to the low arithmetic intensity of processing a single token per forward pass. While dLLMs scale concurrent tokens to increase arithmetic intensity, the integration of MoE reintroduces the memory-bound bottleneck (Figure 1b). This is exacerbated by both algorithmic constraints, where smaller parallel blocks (*e.g.*, 16/32 tokens) are necessary to maintain competitive accuracy [3], and hardware trends, where the FLOPs/byte ratio of modern GPUs continues to outpace memory bandwidth [24]. As a result, MoE dLLM inference remains memory-bound, with latency dominated by loading unique expert weights, a demand that grows far faster than in AR decoding [37].

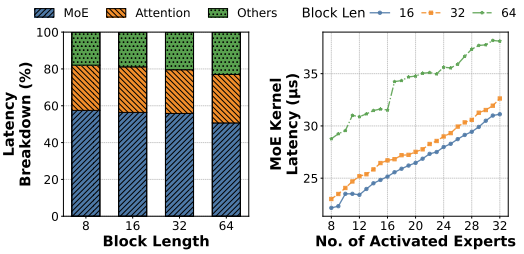


Figure 3: Left: Latency breakdown across MoE FFN, attention, and other components. Right: MoE kernel latency versus activated experts for different block lengths.

Within this regime, the MoE FFN layer emerges as the dominant latency contributor (Figure 3). Following the simplified latency model from previous work [29], the time $f(c)$ to process c tokens through a single expert is $f(c) = ac + b$ for $c > 0$, where b represents the weight fetching cost from HBM to on-chip SRAM and a is the marginal computation cost. The total MoE block latency for a sequence of N parallel tokens is thus:

$$L_{MoE} = \sum_{i=1}^{N_{total}} (b \cdot \mathbb{1}_{cnt_i>0} + a \cdot cnt_i) = b \cdot \left| \bigcup_{n=1}^{N} \mathcal{S}_n \right| + a \cdot (N \cdot K) \tag{1}$$

where cnt_i is the number of tokens routed to expert E_i, and $\mathcal{S}_n$ is the set of K experts activated by the n-th parallel token. As a simple illustration, under uniform routing the expected number of unique activated experts $|\bigcup_{n=1}^{N} \mathcal{S}_n|$ grows as $N_{total}(1 - (1 - K/N_{total})^N)$. Real routers are non-uniform, so measured growth is more gradual (48.1 versus 63.1 experts on LLaDA-MoE 7B at $N = 32$), yet it remains well above DES coreset sizes. As the degree of parallelism N increases, this union set expands rapidly (Figure 1), leading to an increase in latency (Figure 3).

This "**expert explosion**" underscores the limitations of existing expert skipping methods in parallel decoding:

1. **Diminishing Returns from Compute Sparsity:** As latency is dominated by the weight-fetching cost $b \cdot |\bigcup_{n=1}^{N} \mathcal{S}_n|$, reducing the compute term a yields minimal speedup if an expert remains active for any other concurrent token. This motivates "re-activating" experts within a shared coreset to recover accuracy at near-zero marginal latency (Section 5).
2. **Lack of Cross-Token Synergy:** Token-centric skipping ignores the redundancy inherent in parallel decoding. Failing to enforce a collective consensus, they do not minimize the unique expert load $|\bigcup_{n=1}^{N} \mathcal{S}_n|$, leaving the memory bottleneck unresolved.

Optimizing parallel MoE inference thus requires a shift toward *Dynamic Expert Sharing*, which explicitly minimizes the unique expert load $|\bigcup_{n=1}^{N} \mathcal{S}_n|$ at the sequence level.

4 Methodology

4.1 Dynamic Expert Sharing

To optimize the memory-bound bottleneck identified in Section 3, we define a **Coreset Selection Function**, $\Phi : \mathcal{I} \to \mathcal{C}$, which maps available runtime information $\mathcal{I}$ (e.g., aggregated router logits or hidden states) to a shared subset of experts $\mathcal{C} \subset \{E_1, \ldots, E_M\}$. The coreset selection function identifies the most salient experts for a parallel decoding block. Unlike traditional AR batching, where tokens often belong to disparate tasks, parallel decoding tokens share a common context, a unique opportunity for expert reuse. We hypothesize that rather than allowing independent routing,

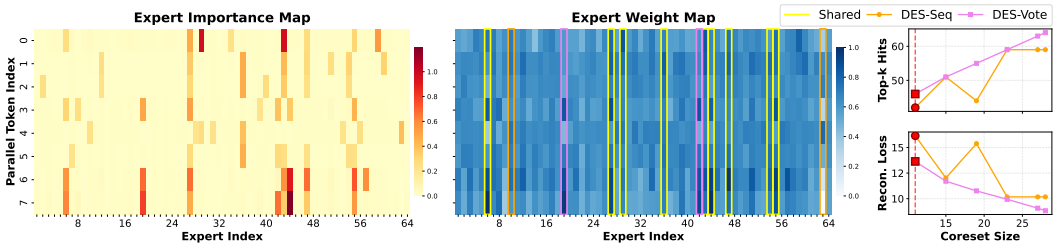


Figure 4: Analysis of Dynamic Expert Sharing (DES) in LLaDA-MoE-7B (8 tokens, Layer 10). (Left) Expert Importance: Heatmap of reconstruction loss sensitivity to expert ablation. Darker regions indicate an increase in reconstruction loss when an expert is removed, revealing strong dependencies between specific token positions and experts. (Middle) Selection Overlays: DES-Seq (orange), DES-Vote (pink), and shared (yellow) selections overlaid on log-routing weights. DES-Vote captures high-importance experts (see index 42) that local ranking misses. (Right) Performance vs. Coreset Size (M_{core}): DES-Vote achieves higher Top-k recall (top) and lower residual reconstruction loss (bottom) than DES-Seq across M_{core}. Red dashed lines denote the specific M_{core} in the heatmaps.

we can identify a compact, sequence-level coreset via cross-token consensus. By restricting each token's Top-k selection to this shared subset, we maximize expert reuse and minimize HBM transfer.

By substituting this coreset into our latency model, we reformulate the MoE layer latency from Equation 1. In the vanilla setting, the unique expert load is defined by the union of independent selections $|\bigcup_{n=1}^{N} \mathcal{S}_n|$, which is implicitly upper-bounded by the total expert pool size M. With the introduction of the coreset, the latency is expressed as:

$$L_{\text{MoE}}(\Phi) \leq b \cdot |\Phi(\mathcal{I})| + a \cdot (N \cdot k) \tag{2}$$

where $|\Phi(\mathcal{I})|$ is the cardinality of the selected coreset. Crucially, whereas the vanilla expert load is constrained by a constant, this formulation transforms the upper bound into an explicit variable $|\Phi(\mathcal{I})|$ that guides the optimization process. We therefore define the search for an optimal coreset selection strategy as a constrained optimization problem:

$$\Phi^* = \arg\min_{\Phi} \quad |\Phi(\mathcal{I})| \text{s.t.} \qquad \mathcal{A}(\Phi(\mathcal{I})) \geq \mathcal{A}_{\text{base}} - \epsilon \tag{3}$$

where $\mathcal{A}(\Phi)$ denotes model accuracy and ϵ is a predefined threshold for tolerable performance degradation. By shifting from independent token-centric routing to collective coreset selection, we decouple the HBM weight-fetching cost from the number of parallel tokens N.

The DES algorithm implements this by transitioning from token-level to sequence-level routing. As detailed in Algorithm 1, the process consists of two stages: (1) **Sequence-level Consensus**, where the shared coreset $\mathcal{C}$ is identified, and (2) **Constrained Local Routing**, where each token selects its Top-k experts exclusively from $\mathcal{C}$. By re-normalizing weights via an activation function σ from the model architecture, we preserve the original routing intent within the optimized memory budget. The algorithm overview is shown in Figure 2.

4.2 Coreset Selection Methods

While the optimization problem in Equation 3 is difficult to solve directly, we approximate the solution by proposing two strategies for dynamic coreset selection: **Intra-Sequence Sharing (DES-Seq)** and **Saliency-Aware Voting (DES-Vote)**. These methods serve as the function $\Phi(\mathcal{I})$ in Algorithm 1 to identify a sequence-wide consensus.

Intra-Sequence Sharing (DES-Seq). A straightforward approach to form a smaller coreset is to select a fixed number of the most salient experts from each token. While previously explored for batch-level optimization in AR models [29], we adapt this to the *intra-sequence* level for dLLM parallel decoding. For each token n in the block, we select its top-k experts, where k is a hyperparameter satisfying $k < K$. The coreset $\mathcal{C}$ is the union of these local selections:

$$\mathcal{C}_{\text{DES-Seq}} = \bigcup_{n=1}^{N} \text{TopK}(\mathcal{I}_n, k) \tag{4}$$

The exact selection algorithm is shown in Algorithm 2.

Algorithm 1 Dynamic Expert Sharing (DES)

Require: Sequence info $\mathcal{I}$, Coreset selector Φ, Activation σ, Target K.

// Stage 1: Sequence-level Consensus
$\mathcal{C} \leftarrow \Phi(\mathcal{I})$ {Identify expert coreset}
// Stage 2: Constrained Local Routing
for each token $n \in \{1, \ldots, N\}$ **do**
  $\mathcal{S}_n \leftarrow \text{TopK}(\mathcal{I}_n|_{i \in \mathcal{C}}, K)$ {Route in coreset}
  $g_n \leftarrow \sigma(\mathcal{I}_n|_{i \in \mathcal{S}_n})$ {Re-norm. gates}
  $y_n \leftarrow \sum_{i \in \mathcal{S}_n} g_{n,i} \cdot E_i(x_n)$
end for
return $Y = \{y_1, \ldots, y_N\}$

Algorithm 2 DES-Seq: Intra-Sequence Sharing

Require: Router logits $\mathcal{I}$, Threshold k.
$\mathcal{C} \leftarrow \bigcup_{n=1}^{N} \text{TopK}(\mathcal{I}_n, k)$
return $\mathcal{C}$

Algorithm 3 DES-Vote: Saliency-Aware Voting

Require: Router logits $\mathcal{I}$, Size M_{core}, Top-K.
$\mathcal{I}_m \leftarrow \text{Mask}(\mathcal{I}, K)$ {Top-K gates}
$V \leftarrow \sum_{n=1}^{N} \mathcal{I}_{m,n}$ {Aggregate}
$\mathcal{C} \leftarrow \text{TopK}(V, M_{\text{core}})$ {Pick coreset}
return $\mathcal{C}$

Saliency-Aware Voting (DES-Vote). Despite its simplicity, DES-Seq has two primary limitations. First, it does not explicitly maximize expert sharing; it merely reduces local budgets without seeking a global consensus. Second, it utilizes a fixed selection threshold k for all tokens, ignoring that expert importance varies significantly across a sequence (e.g., the 2nd-ranked expert for token A might be more critical than the 2nd-ranked expert for token B).

To address these, we propose **Saliency-Aware Voting (DES-Vote)**. To maximize consensus, tokens vote for a collective set. A naive approach is uniform voting for each token's top-k experts. However, as seen in the **Expert Importance Map** and **Expert Weight Map** (Figure 4), there is a high correlation between raw gating weights and actual importance. We hypothesize that assigning vote weights based on router scores is effective, as it accounts for varying influences within the top-k.

In practice, we aggregate the router weights across the sequence, but crucially **mask out** weights of experts falling outside each token's local top-k selection to filter noise, and renormalize the remaining weights into each token's top-k gates. This addresses DES-Seq's second limitation by allowing the collective importance to naturally dictate which experts are retained. The coreset $\mathcal{C}$ is formed by selecting the top M_{core} experts by total vote:

$$V_i = \sum_{n=1}^{N} \text{Masked}(\mathcal{I}_{n,i}), \quad \mathcal{C}_{\text{DES-Vote}} = \text{TopK}(V, M_{\text{core}}) \tag{5}$$

As shown in Figure 4, DES-Vote outperforms DES-Seq in both top-K hit rate (preserving more ground truth selections) and reconstruction loss across varying coreset sizes.

4.3 Custom Kernel for Coreset Selection

To mitigate the system overhead of fragmented operator execution, we develop a custom fused GPU kernel that collapses 12 kernels (e.g., softmax, top-k, and reduction) into just two. This design addresses the kernel launch and memory traffic bottlenecks. The primary kernel fuses per-token softmax and top-k filtering with weighted expert accumulation, utilizing register-level computation and atomic instructions to update global saliency scores efficiently. A second kernel then performs final expert masking based on a threshold-governed ranking.

4.4 Theoretical Guarantees

We now bound the MoE output perturbation caused by restricting routing to a coreset $\mathcal{C}$, and show that DES-Vote minimizes this bound. For token n, let $\mathcal{S}_n$ be its vanilla top-K set and $\tilde{\mathcal{S}}_n(\mathcal{C})$ its top-K set restricted to $\mathcal{C}$. Let g_n and $\tilde{g}_n(\mathcal{C})$ be the corresponding gate vectors over all M experts, obtained by renormalizing the same nonnegative router scores over $\mathcal{S}_n$ and $\tilde{\mathcal{S}}_n(\mathcal{C})$, and let y_n and $\tilde{y}_n(\mathcal{C})$ be the resulting MoE outputs. We define the **discarded routing mass** as $\varepsilon_n(\mathcal{C}) = \sum_{i \in \mathcal{S}_n \setminus \mathcal{C}} g_{n,i}$. Throughout, we assume $\|E_i(x_n)\| \le B$ for all experts, $|\mathcal{C}| \ge K$, and consistent tie-breaking.

Theorem 4.1 (Per-token certificate)**.** *For every token n, $\|y_n - \tilde{y}_n(\mathcal{C})\| \le 2B\,\varepsilon_n(\mathcal{C})$. In particular, $\varepsilon_n(\mathcal{C}) = 0$ implies $\tilde{y}_n(\mathcal{C}) = y_n$.*

Proof. Since $\mathcal{S}_n$ is the global top-K set, every expert in $\mathcal{S}_n \cap \mathcal{C}$ remains in $\tilde{\mathcal{S}}_n(\mathcal{C})$. Replacing excluded experts with lower-scoring ones cannot increase the normalization denominator, hence $\tilde{g}_{n,i}(\mathcal{C}) \geq g_{n,i}$ for all $i \in \mathcal{S}_n \cap \mathcal{C}$. Excluded experts lose exactly $\varepsilon_n(\mathcal{C})$ mass, and as both gate vectors sum to one, surviving and replacement experts gain exactly the same mass, so $\|g_n - \tilde{g}_n(\mathcal{C})\|_1 = 2\varepsilon_n(\mathcal{C})$. The bound follows from $y_n - \tilde{y}_n(\mathcal{C}) = \sum_i (g_{n,i} - \tilde{g}_{n,i}(\mathcal{C}))E_i(x_n)$ and $\|E_i(x_n)\| \leq B$. If $\varepsilon_n(\mathcal{C}) = 0$, then $\mathcal{S}_n \subseteq \mathcal{C}$, so $\tilde{\mathcal{S}}_n(\mathcal{C}) = \mathcal{S}_n$. □

Theorem 4.2 (Optimality of DES-Vote)**.** *Let $V_i = \sum_{n=1}^{N} g_{n,i}$ be the vote of expert i in Equation 5. For any coreset with $|\mathcal{C}| = M_{core}$, $\sum_{n=1}^{N} \varepsilon_n(\mathcal{C}) = \sum_{i \notin \mathcal{C}} V_i$. Therefore, $\mathcal{C}_{DES\text{-}Vote}$ minimizes the summed certificate of Theorem 4.1 over all coresets of size M_{core}.*

Proof. As $g_{n,i} = 0$ for $i \notin \mathcal{S}_n$, we have $\varepsilon_n(\mathcal{C}) = \sum_{i \notin \mathcal{C}} g_{n,i}$. Exchanging the sums over n and i gives the identity, which is minimized by keeping the M_{core} largest votes. □

Corollary 4.3 (Arbitrary routing heterogeneity)**.** *Let $V_{[1]} \geq \cdots \geq V_{[M]}$ be the sorted votes and $\beta = M_{core}/M$. For $\mathcal{C} = \mathcal{C}_{DES\text{-}Vote}$,*

$$\frac{1}{N}\sum_{n=1}^{N} \|y_n - \tilde{y}_n(\mathcal{C})\| \leq 2B\Big(1 - \frac{1}{N}\sum_{j=1}^{M_{core}} V_{[j]}\Big) \leq 2B(1-\beta) \tag{6}$$

Proof. The first inequality follows from Theorems 4.1 and 4.2 with $\sum_i V_i = N$. The second holds as the M_{core} largest of M nonnegative votes carry at least a β fraction of their sum. □

The first bound in Equation 6 is a data-dependent runtime certificate, and the second holds even when the candidate union $\bigcup_{n=1}^{N} \mathcal{S}_n$ covers all M experts.

5 Experiments

5.1 Experimental Setup

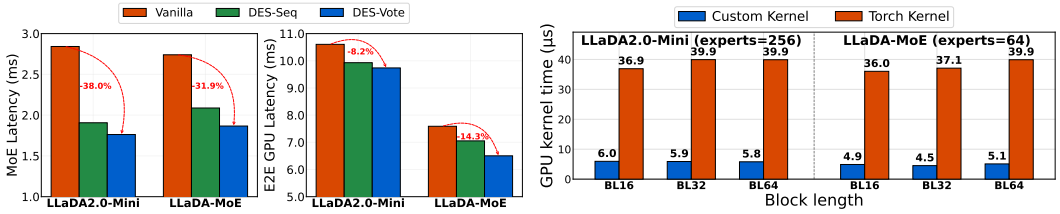
Implementations. We evaluate two frontier MoE dLLMs for parallel decoding: the 16B **LLaDA2.0-mini-preview** [5] and 7B **LLaDA-MoE-7B-A1B-Instruct** [39]. Our experiments utilize the dInfer framework [25] with Fast-dLLM [34] as the KV cache method, adopting a 0.9 confidence-based sampling threshold and default hyperparameter configurations. All experiments were conducted on **NVIDIA B200 GPUs** [27] with CUDA 13.1, paired with an Intel Xeon 6960P CPU. Kernel time was profiled using the NVIDIA Nsight Systems toolkit [28]. For DES, we parameterize DES-Vote using a budget factor β such that the coreset size $M_{\text{core}} = \beta \times M$ (where M is the total expert pool size), while DES-Seq is controlled by the local selection count k. We treat M_{core} as the primary resource budget: results use the same two sizes, $M_{\text{core}} \in \{38, 25\}$, across all tasks and both models (i.e., $\beta \approx 0.15, 0.10$ for the 256 experts of LLaDA2.0-Mini and $\beta \approx 0.60, 0.40$ for the 64 experts of LLaDA-MoE 7B), requiring no per-task or per-dataset search. In deployment, M_{core} can be set directly from the available expert memory or latency budget; if a specific accuracy target is required, it is a one-dimensional parameter that can be tuned on a small held-out set with efficient methods [30, 18, 12].

Datasets. The algorithm performance is assessed across four benchmarks requiring long-form generative decoding and diverse reasoning: HumanEval [8], MBPP [4], GSM8K [11], and MATH500 [20].

Baselines. As existing expert skipping baselines were primarily designed for AR models, we re-implement them for parallel dLLM decoding. Specifically, we adapt **NAEE** [23] by skipping the bottom experts i through K if their cumulative probability $\sum_{u=i}^{K} \pi_{top-u}$ falls below a relative threshold β of the total routing sum. We calibrate β using the frontier search approach from **(author?)** [17]. Additionally, we also compare with **MC-MoE** [16], which preserves the experts of the important tokens. Notably, we adopt only the dynamic expert skipping components of these baselines, ensuring a fair comparison. Finally, we compare against a **Top-K** baseline, which reduces expert traffic by directly decreasing the K value in expert selection.

Table 1: Main results on generative benchmarks for LLaDA2.0-Mini and LLaDA-MoE-7B. $\mathcal{T}$ denotes the average number of unique activated experts per layer. R.Acc denotes accuracy relative to the **Vanilla** model. *Mem.* is the average memory footprint of unique activated MoE parameters per layer. All experiments use a block length of 32 with 16 prefix and 16 suffix cache tokens. Best results are **bolded**; the superior performer in adjacent DES-Seq vs. DES-Vote pairs is underlined.

Method	MBPP		GSM8K		HumanEval		MATH500		Avg. $\mathcal{T}$	Mem.	Avg. R. Acc
	$\mathcal{T}$	R. Acc.	$\mathcal{T}$	R. Acc.	$\mathcal{T}$	R. Acc.	$\mathcal{T}$	R. Acc.		GB	%
LLaDA2.0-Mini 16B											
Vanilla	78	100.0	87	100.0	84	100.0	85	100.0	84	0.98	100.0
top-k ($k = 4$)	52	70.5	58	53.8	56	91.4	58	83.6	56	0.66	74.8
NAEE ($\beta = 0.6$)	60	44.0	67	61.6	67	20.5	66	60.5	65	0.76	46.6
MC-MOE ($\beta = 0.6$)	61	45.9	67	60.2	68	18.0	67	62.7	66	0.77	46.7
DES-Seq ($k = 3$)	41	100.4	46	**99.7**	45	92.4	46	96.5	45	0.53	97.2
DES-Vote (M_{core}=38)	38	**104.7**	38	98.4	38	97.5	38	**97.4**	38	0.45	**99.5**
DES-Seq ($k = 2$)	31	95.1	35	93.9	34	**100.0**	35	93.6	34	**0.40**	95.7
DES-Vote (M_{core}=25)	**25**	96.4	**25**	95.0	**25**	97.5	**25**	96.5	**25**	0.29	96.4
LLaDA-MoE 7B											
Vanilla	59	100.0	59	100.0	59	100.0	59	100.0	59	0.35	100.0
top-k ($k = 4$)	46	85.6	48	80.6	48	79.4	48	78.9	48	0.28	81.1
NAEE ($\beta = 0.6$)	50	96.6	50	96.6	51	99.0	50	93.5	50	0.29	96.4
MC-MOE ($\beta = 0.6$)	52	84.8	53	93.5	54	79.4	53	68.2	53	0.31	81.5
DES-Seq ($k = 3$)	41	**98.7**	42	99.1	42	96.1	42	93.9	42	0.25	96.9
DES-Vote (M_{core}=38)	38	97.9	38	**100.1**	38	95.0	38	**97.3**	38	0.22	**97.6**
DES-Seq ($k = 2$)	33	**98.5**	34	98.8	34	96.1	34	88.1	34	0.20	95.4
DES-Vote (M_{core}=25)	**25**	95.6	**25**	98.0	**25**	**101.0**	**25**	89.3	**25**	**0.15**	96.0



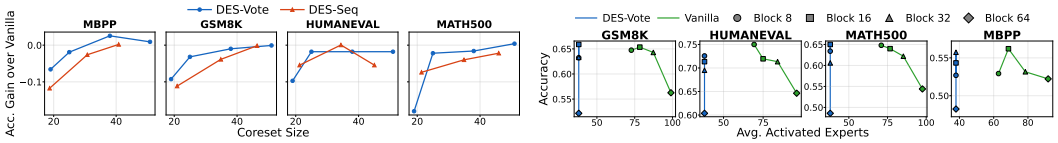
(a) MoE and end-to-end latency across models.

(b) Coreset selection kernel profiling.

Figure 5: **(a)** Latency measurements for the MoE kernel (left) and total end-to-end GPU kernel execution time (right) across models. **(b)** Profiling of coreset selection latency: our custom fused kernel achieves a large speedup over the PyTorch baseline.

5.2 Main Results

As shown in Table 1, existing expert skipping methods such as NAEE and MC-MoE suffer from severe degradation for dLLMs, retaining only ∼46% accuracy on LLaDA2.0-Mini. This failure stems from the suboptimality of applying static skipping thresholds across parallel tokens with diverse gating distributions as shown in Figure 4, which often leads to over-sparsification (see Appendix B for NAEE calibration). In contrast, DES improves performance by exploring dynamic sequence-level expert sharing. On LLaDA2.0-Mini, DES-Vote maintains a 99.5% relative accuracy compared to vanilla while reducing the unique expert load by 55%, demonstrating an effective approximation of the optimization objective in Equation 3. Notably, incorporating local saliency (MC-MoE) yields negligible improvements over NAEE, suggesting that token-centric metrics are insufficient to address the collective redundancy inherent in parallel decoding. Finally, DES-Vote outperforms DES-Seq by achieving higher average accuracy with a smaller expert footprint. The relative accuracy of DES-Vote surpasses that of DES-Seq from 0.6% to 2.3%, consistently with lower active experts. These results validate that collective voting of DES-Vote effectively identifies a high-utility coreset that captures expert saliency across the sequence.



(a) Accuracy gain across varying coreset sizes.

(b) Accuracy vs. active experts across block sizes.

Figure 6: Analysis of efficiency-accuracy trade-offs. All experiments are done with LLaDA2.0-Mini.

5.3 Hardware Efficiency Analysis

We evaluate the hardware efficiency of DES on a single NVIDIA B200 GPU. As shown in Figure 5a, DES-Vote reduces MoE layer latency by up to 38.0% for LLaDA2.0-Mini and 31.9% for LLaDA-MoE-7B. Total end-to-end GPU kernel time improves by 8.2–14.3%. The expected dilution in relative gains stems from constant non-MoE operations like self-attention. These results confirm that by mitigating sequence-level HBM traffic, DES provides substantial wall-clock savings across the entire inference pipeline. As shown in Figure 5b, our fused kernel (Sec. 4.3) achieves a 6× speedup in coreset selection by eliminating redundant HBM traffic and operator dispatch overhead.

5.4 Scalability Analysis

Batched serving. We extend our analysis to practical online serving that batches multiple requests for higher throughput. Table 2 evaluates LLaDA2.0-Mini under homogeneous (MBPP) and heterogeneous (MBPP+GSM8K) batches. Relative latency gains *grow* with batch size and peak in the moderate range (BS=2–4) reaching up to 41% MoE-kernel and 23% E2E latency reduction. Mixing tasks inflates vanilla expert counts but DES still compresses them effectively. BS=16 OOMs because dLLMs lack cross-request prefix-cache sharing, placing realistic deployments in the batch size regime where DES delivers the most value. DES operates along the sequence dimension and is therefore orthogonal to batch-level methods like OEA [29].

Table 2: Batched serving of LLaDA2.0-Mini on a single B200 (β=0.10).

BS	Vanilla			DES-Vote		
	$\mathcal{T}$	MoE Lat. (ms)	E2E Lat. (ms)	$\mathcal{T}$	MoE Lat. (ms)	E2E Lat. (ms)
Homogeneous (MBPP)						
1	63.6	8.6	19.4	25.0	5.8	17.5
2	74.1	9.4	20.7	27.9	5.7	17.6
4	77.3	10.2	23.8	28.4	6.0	19.2
8	79.0	20.6	81.5	30.1	15.2	66.3
Heterogeneous (MBPP + GSM8K)						
2	79.3	9.1	18.8	27.9	4.6	14.6
4	81.1	9.9	22.4	30.7	5.9	19.3
8	83.3	16.1	60.8	33.0	14.2	56.4

Scaling up. To verify DES at larger scale, we evaluate the 103B **LLaDA2.0-flash** on MBPP (Table 3). DES-Vote (β=0.10) compresses the active footprint by 63% at no accuracy loss, and a more aggressive β=0.05 shrinks the set to just 12 experts at 96.4% relative accuracy. Appendix A further studies the performance on expert parallelism (EP).

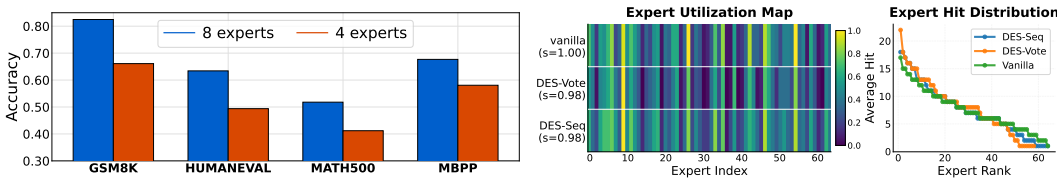
Table 3: DES on the 103B **LLaDA2.0-flash** on 4×B200 with tensor parallelism (MBPP).

Method	$\mathcal{T}$	R. Acc. (%)	MoE Lat. (ms)
Vanilla	67.9	100.0	5.7
DES-Vote (β=0.10)	25.0	100.0	5.1
DES-Vote (β=0.05)	12.0	96.4	4.9

5.5 Ablation Studies

Effect of coreset size. Figure 6a, model performance generally correlates positively with the coreset size, gradually declining as coreset shrinks. Across all benchmarks, DES-Vote consistently maintains higher accuracy than DES-Seq at similar coreset sizes. Furthermore, by leveraging the continuous β, DES-Vote offers greater flexibility in modulating coreset size, enabling extremely small coresets that bypass the one-expert-per-token lower bound inherent to DES-Seq.

Robustness to block sizes. Figure 6b demonstrates the performance of DES-Vote across varying parallel block sizes ($\{8, 16, 32, 64\}$). The accuracy drop remains consistently small across different tasks as the block size increases, indicating that the coreset selected from DES-Vote's collective voting effectively generalizes to larger degrees of parallelism. This shows that DES-Vote maintains a constant and low count of activated experts regardless of block size. In contrast, the vanilla model suffers from a sharp increase in expert activations as the block size grows, leading to a severe memory



(a) Effect of active expert count on LLaDA2.0-mini. (b) Expert utilization on LLaDA-MoE 7B (MBPP).

Figure 7: **(a)** Both configurations activate the same number of unique experts, differing only in per-token expert computation ($k = 8$ vs. $k = 4$). **(b) (Left)** Normalized hit-rate heatmaps across a parallel block, where s denotes the cosine similarity between the coreset and vanilla expert hit rate vectors. **(Right)** Expert activation frequencies obtained by ranking experts by their average hit counts within each method. Results are on the 10th layer of LLaDA-MoE 7B.

traffic bottleneck. This decoupling of memory overhead from parallelism fundamentally shifts the design space for parallel decoding like dLLMs. While recent research emphasizes that block size is vital for balancing algorithmic performance and throughput [22], DES-Vote effectively neutralizes the associated efficiency penalties. Consequently, practitioners can optimize block size based purely on the trade-off between multi-token generation efficiency and model accuracy, unconstrained by the traditional limits of memory bandwidth.

Effect of active expert count. Figure 7a compares the performance of DES with different numbers of experts per token. Specifically, both configurations identify an equally sized, compact set of experts (Algorithm 1, Step 2), hence differing only in their per-token expert computation (Algorithm 1, Step 5). Notably, reducing per-token computation from 8 to 4 experts, even while activating the same total number of unique experts, results in a consistent and significant accuracy decrease across all benchmarks. This confirms that re-activating experts from the coreset can preserve performance even when these experts differ from the top-8 originally selected by the vanilla model.

5.6 Visualization of Expert Utilization

To evaluate how effectively our methods preserve the model's original routing logic, we analyze expert activation patterns in Figure 7b. First, both DES-Seq and DES-Vote exhibit high cosine similarity ($s \geq 0.98$) with the vanilla expert hit rate vector. This high representational fidelity confirms that restricting the expert pool to a coreset does not distort the model's fundamental routing pattern. Second, the expert hit distribution reveals a concentration effect. The activation curve for DES-Vote is noticeably sharper than that of DES-Seq and the vanilla baseline. DES-Vote mitigates the long tail of expert hit distribution, hence reducing memory traffic with concentrated expert activation.

6 Conclusion

In this paper, we characterize the *expert explosion* phenomenon in dLLM MoE decoding, where the weight-fetching overhead scales linearly with the parallel block size. By identifying a compact, high-utility expert coreset through two strategies, DES-Seq and DES-Vote, our method effectively decouples HBM memory traffic from the degree of parallelism. Extensive experiments demonstrate that DES significantly reduces unique expert activations and MoE latency while retaining accuracy. As memory bandwidth increasingly lags behind processor speed and model expert sparsity grows, we show that DES is a viable approach for realizing the high-throughput potential of parallel generation, motivating further research into sequence-level expert sharing for parallel decoding models.

Limitations. DES is not lossless, and its accuracy–efficiency trade-off varies across tasks and compression levels. The achievable compression and the appropriate budget are model-dependent, and end-to-end gains are smaller than MoE kernel gains since DES does not accelerate attention.

Acknowledgments and Disclosure of Funding

This work was conducted during Hao (Mark) Chen's internship at Samsung AI Center, Cambridge, UK. Hao was generously sponsored by the President's Scholarship from Imperial College London.

References

[1] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[2] 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.

[3] 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. *arXiv preprint arXiv:2503.09573*, 2025.

[4] 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.

[5] 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.

[6] Shiyi Cao, Shu Liu, Tyler Griggs, Peter Schafhalter, Xiaoxuan Liu, Ying Sheng, Joseph E Gonzalez, Matei Zaharia, and Ion Stoica. Moe-lightning: High-throughput moe inference on memory-constrained gpus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 715–730, 2025.

[7] Hao Mark Chen, Wayne Luk, Ka Fai Cedric Yiu, Rui Li, Konstantin Mishchenko, Stylianos I Venieris, and Hongxiang Fan. Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference. *arXiv preprint arXiv:2405.18628*, 2024.

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

[9] 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.

[10] 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.

[11] 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.

[12] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.

[13] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

[14] Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. Set block decoding is a language model inference accelerator. *arXiv preprint arXiv:2509.04185*, 2025.

[15] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[16] Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts llms gains more. *ICLR*, 2025.

[17] 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.

[18] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[19] Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334*, 2023.

[20] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[21] Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305, 2025.

[22] Guanxi Lu, Hao Mark Chen, Yuto Karashima, Zhican Wang, Daichi Fujiki, and Hongxiang Fan. Adablock-dllm: Semantic-aware diffusion llm inference via adaptive block size. *arXiv preprint arXiv:2509.26432*, 2025.

[23] 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. *ACL*, 2024.

[24] Xiaoyu Ma and David Patterson. Challenges and research directions for large language model inference hardware. *arXiv preprint arXiv:2601.05047*, 2026.

[25] Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, et al. dinfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv:2510.08666*, 2025.

[26] Bradley McDanel, Steven Li, Sruthikesh Surineni, and Harshit Khaitan. Moe-spec: Expert budgeting for efficient speculative decoding. *arXiv preprint arXiv:2602.16052*, 2026.

[27] NVIDIA Corporation. Nvidia blackwell architecture technical brief. Technical report, NVIDIA Corporation, 2025.

[28] NVIDIA Corporation. Nvidia nsight systems, 2026.

[29] 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.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

[31] Daniil Vankov, Nikita Ivkin, Kyle Ulrich, Xiang Song, Ashish Khetan, and George Karypis. Xshare: Collaborative in-batch expert sharing for faster moe inference. *arXiv preprint arXiv:2602.07265*, 2026.

[32] 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.

[33] Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025.

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

[35] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[36] 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.

[37] Yanpeng Yu, Haiyue Ma, Krish Agarwal, Nicolai Oswald, Qijing Huang, Hugo Linsenmaier, Chunhui Mei, Ritchie Zhao, Ritika Borkar, Bita Darvish Rouhani, et al. Efficient moe serving in the memory-bound regime: Balance activated experts, not tokens. *arXiv preprint arXiv:2512.09277*, 2025.

[38] Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 86–102, 2025.

[39] Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, et al. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025.

A Expert Parallelism

While our main experiments isolate algorithmic effects on a single GPU, large-scale MoE deployment commonly partitions experts across devices via Expert Parallelism (EP). EP introduces a new dimension, an irregular all-to-all dispatch of tokens to remote experts, on top of the SRAM-to-HBM weight loading bottleneck. DES is well-suited to both directions: by enforcing a sequence-level consensus, it (i) directly reduces unique expert traffic in the memory-bound decode regime [37], and (ii) shrinks the all-to-all communication volume because fewer distinct experts are addressed in each forward pass.

We empirically verify these benefits using LLaDA2.0-Mini on MBPP under two distributed setups (Table 4). On 2×NVIDIA B6000, DES-Vote cuts the fused MoE kernel latency by 1.76×; on 4×NVIDIA B200, where the per-GPU expert footprint is already small, DES still delivers a 1.10× MoE-kernel speedup.

Table 4: Fused MoE kernel latency under EP for LLaDA2.0-Mini on MBPP. DES-Vote uses $\beta = 0.10$.

Method	**2× B6000 (EP=2)**		**4× B200 (EP=4)**	
	MoE (ms)	**Speedup**	**MoE** (ms)	**Speedup**
Vanilla	5.1	—	1.4	—
DES-Vote	2.9	1.76×	1.3	1.10×

B NAEE Calibration

To verify that the NAEE result on LLaDA2.0-Mini is not caused by one poorly chosen β, we rerun NAEE on the full MATH500 and MBPP datasets, sweeping flat β and reproducing its 25pct and 50pct layerwise calibration protocols (Table 5). Calibration improves NAEE only at a substantially larger footprint: the 25pct setting activates 58–72% more experts than DES-Vote while remaining 9.9–28.0 relative-accuracy points lower. At the tighter 50pct setting, NAEE still activates more experts than DES-Vote but its accuracy collapses. Flat thresholds perform even worse. This is consistent with their mechanisms: NAEE removes experts independently from each token, so the total number of experts in the block is not explicitly minimized.

Table 5: NAEE sweep and calibration on LLaDA2.0-Mini. Accuracy is relative to **Vanilla**.

Setting	**MATH500**		**MBPP**	
	$\mathcal{T}$	**R. Acc.** (%)	$\mathcal{T}$	**R. Acc.** (%)
NAEE, flat $\beta = 0.2$	69.5	39.5	63.0	15.4
NAEE, flat $\beta = 0.35$–0.70	≤59.5	≤1.3	≤52.2	0.0
NAEE, calibrated 25pct	65.5	87.5	≈60	76.7
NAEE, calibrated 50pct	52.9	4.0	≈44	1.3
DES-Vote (M_{core}=38)	38	97.4	38	104.7

C Additional Theoretical Remarks

Remark C.1 (Safeguard). Aggregate optimality in Theorem 4.2 can still sacrifice a niche token in a highly heterogeneous block. As $\varepsilon_n(\mathcal{C})$ is computable from router outputs before any expert executes, one could enforce $\max_n \varepsilon_n(\mathcal{C}) \le \tau$ by enlarging $\mathcal{C}$ with the highest-weight omitted experts of violating tokens, or by falling back to vanilla routing, guaranteeing $\|y_n - \tilde{y}_n(\mathcal{C})\| \le 2B\tau$ for every token. We do not evaluate this safeguard in Section 5.

Remark C.2 (Scope). The bounds in Section 4.4 are per-layer bounds on MoE output perturbation. A nontrivial bound on discrete benchmark accuracy is impossible without a downstream stability assumption, and composing Lipschitz constants through multiple Transformer blocks makes such a bound very loose. We therefore validate accuracy empirically in Section 5.

D Societal Impact

By mitigating the expert explosion problem, our approach contributes to more energy-efficient and cost-friendly dLLM serving. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

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: See Section 1 and 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: See the Limitations paragraph in Section 6.

   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: [Yes]

Justification: See Section 4.4.

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: See Section 4 and Section 5.

   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: See supplementary materials.

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: See Section 5.

   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: [No]

   Justification: We follow the prevailing practice for LLM benchmarking and report point estimates from deterministic decoding runs with fixed seed, as 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: See Section 5.

   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: We have reviewed the NeurIPS Code of Ethics and conformed to it.

   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: See Appendix D.

    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 does not release any new models or datasets that pose a risk of misuse.

    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: See Section 5.

    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: [N/A]

    Justification: The paper does not release any new datasets, models, or pretrained checkpoints.

    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: The paper does not involve crowdsourcing or 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: The paper does not involve 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: LLMs are the subject of study rather than a non-standard component of our methodology.

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.