PHASE-MOE: A CO-DESIGN TO BOUND THE EXPERT EXPLOSION IN BLOCK DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Block Diffusion Language Models (BDLMs) have emerged as a compelling paradigm bridging autoregressive modeling and discrete diffusion, pairing inter-block causal generation with intra-block bidirectional parallel decoding. To scale these architectures efficiently, integrating Mixture-of-Experts (MoE) is essential. However, combining MoE with block-parallel decoding triggers a systemic *Expert Explosion*: processing multiple tokens simultaneously causes their independently routed experts to span the expert pool. This inflates memory traffic from High-Bandwidth Memory (HBM) to SRAM, creating a memory-bound bottleneck. We demonstrate that existing post-hoc mitigations break down under production continuous batching, where the temporal and semantic heterogeneity of asynchronous requests escalates global memory traffic to near-dense levels. To resolve this, we propose a bandwidth-aware co-design comprising *Phase-Constrained MoE* and a *Phase-Aware Scheduler*. Recognizing that discrete diffusion progresses through predictable mask-density phases, we condition the router's expert pools on the temporal denoising phase during training, leaving compute-bound prefill unconstrained. The *Phase-Aware Scheduler* subsequently clusters asynchronous requests by mask density at runtime, promoting overlapping expert execution. Under production-scale continuous batching simulations where existing mitigations fail, our co-design reduces unique expert activations by up to 56.2% and cuts MoE kernel latency by up to 38.5%, preserving competitive generative performance.

1 INTRODUCTION

Efficient sequence generation at scale fundamentally hinges on co-designing expressive architectures with the physical bandwidth realities of modern memory hierarchies.

Block Diffusion and Acceleration. Autoregressive (AR) large language models dominated sequence generation, but their left-to-right causal dependency imposes a severe, memory-bound bottleneck during decoding (Pope et al., 2023; Kwon et al., 2023a), alongside fundamental limitations inherent to unidirectional causal generation, such as the reversal curse (Berglund et al., 2024; Wang & Sun, 2025; Golovneva et al., 2024; Lu et al., 2024). Diffusion language models (DLMs) stand out as a compelling alternative, reformulating generation as an iterative denoising process over discrete variables (see Appendix A.2 for formal definitions and background) (Austin et al., 2021a; Hoogeboom et al., 2021; Campbell et al., 2022; Lou et al., 2023; Sahoo et al., 2024b; Nie et al., 2025; Gong et al., 2024; Bie et al., 2025; Zhu et al., 2025). To unite causal KV-caching efficiency with intra-block bidirectional decoding, Block Diffusion Language Models (BDLMs) serve as an effective paradigm (Arriola et al., 2025; Fathi et al., 2025), often leveraging pre-trained AR initializations (Bie et al., 2025; Taïga et al., 2026; Wu et al., 2025a; Cheng et al., 2025b). By predicting multiple masked tokens simultaneously, BDLMs unlock parallel decoding capabilities. Recent research has focused on accelerating this iterative decoding through caching mechanisms, dynamic block sizing, and speculative verification (Wu et al., 2025a;b; 2026; Qi et al., 2026; Hong et al., 2025; Lu et al., 2025).

The MoE Scaling Paradigm. To scale the capacity of these architectures without incurring prohibitive computational costs, integrating Mixture-of-Experts (MoE) is imperative (Jiang et al., 2024; Du et al., 2022; Zoph et al., 2022; Lepikhin et al., 2020; Shazeer et al., 2017; Fedus et al., 2022). Recent advancements in large-scale LLMs have refined MoE routing via fine-grained experts, auxiliary-loss-free balancing, and expert-router coupling (Dai et al., 2024; Wang et al., 2024; Lv et al., 2026;

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.

Guo et al., 2026; Lv et al., 2025), demonstrating that careful sparsity management is critical for reasoning tasks (Zhao et al., 2024; Abnar et al., 2025; Nakamura et al., 2026). In the diffusion domain, MoE has been successfully applied to both continuous and discrete models—predominantly adopting token-level Top-k routing—to accelerate convergence and increase parameter efficiency (Taïga et al., 2026; Bie et al., 2025; Zhu et al., 2025; Liu et al., 2025; Cheng et al., 2025a), largely overlooking the resulting inference-time memory traffic. Beyond standard token-choice selection, alternative routing paradigms, such as expert-choice routing, have also been explored across vision and language diffusion to leverage non-causal global context (Zhang et al., 2026; Sun et al., 2025; Shi et al., 2025).

The Expert Explosion. Despite these advances, combining MoE with the parallel block decoding of BDLMs triggers a system-level bottleneck: the *Expert Explosion*. During AR next-token decoding, each step activates a bounded subset of experts. In contrast, parallel processing of B tokens activates the union of their independently routed experts, which scales to exhaust global expert capacity (Taïga et al., 2026; Wei et al., 2026; Feng et al., 2026; Chen et al., 2026). This pushes inference into a High-Bandwidth Memory (HBM) bottleneck—where generation stalls on transferring parameters from memory (Chen et al., 2026; Williams et al., 2009), negating parallel decoding speedups.

The Continuous Batching Bottleneck. Recent post-hoc mitigations—whether deploying inference heuristics like TEAM (Wei et al., 2026) and DES (Chen et al., 2026), or lightweight distillation like dMoE (Feng et al., 2026)—are designed exclusively for isolated, single-sequence execution. When extended to multi-sequence continuous batching (Yu et al., 2022; Kwon et al., 2023b; Patel et al., 2024; Agrawal et al., 2024; 2023), these methods encounter severe design trade-offs that we formalize in Section 2 and validate empirically in Section 5.3, motivating a structural, training-time resolution.

Phase Heterogeneity. Discrete diffusion traverses predictable denoising phases: early stages (high mask density) govern global semantic planning, while late stages (low mask density) focus on local syntactic refinement. Prior studies confirm that diffusion models exhibit pronounced temporal heterogeneity and shifting expert utilization across timesteps (Park et al., 2024b;a; Lee et al., 2024; Balaji et al., 2022), alongside phase-dependent decoding dynamics across token confidence, convergence, and refinement oscillations (Kim et al., 2025; Li et al., 2025; Wang et al., 2026).

Contributions. To resolve the Expert Explosion under multi-sequence continuous batching, we propose an algorithm–system co-design that grounds memory-bandwidth bounds into training-time architectural guarantees and aligns asynchronous requests at runtime:

1. **Formalization of the Continuous Batching Dilemma:** We analytically define the *Expert Explosion* in MoE-based BDLMs and demonstrate that existing post-hoc routing mitigations face a fundamental dilemma under multi-sequence temporal and semantic heterogeneity (Section 2).
2. **Phase-Constrained MoE:** We introduce a training-time architectural guarantee that explicitly bounds candidate expert subsets to specific denoising phases via a novel soft-to-hard optimization curriculum, while isolating compute-bound prefill (Section 3).
3. **Phase-Aware Scheduling:** We design an inference-time dispatch policy that clusters asynchronous requests by mask density. This maximizes overlapping expert execution without tail-end starvation, effectively translating training-time priors into hardware efficiency (Section 4).
4. **Breaking the Memory-Accuracy Trade-off:** In continuous batching benchmarks, our co-design slashes unique expert activations by up to 56.2% and kernel latency by up to 38.5%, sustaining throughput where existing post-hoc methods break down (Section 5).

2 Problem Formulation: The Continuous Batching Dilemma

BDLMs (Bie et al., 2025; Arriola et al., 2025; Fathi et al., 2025; Taïga et al., 2026; Wu et al., 2025a; Cheng et al., 2025b) (detailed in Appendix A) iteratively denoise blocks of B tokens (where B_{step} denotes the sequence batch size in continuous batching). While this unlocks semi-autoregressive parallel decoding, scaling these architectures via MoE introduces a system-level bottleneck.

The Expert Explosion. In an MoE layer, execution latency T_{MoE} is governed by both memory bandwidth and arithmetic compute (Chen et al., 2026; Oncescu et al., 2025):

$$T_{\text{MoE}} = c_{\text{mem}} \cdot \left| \bigcup_{i=1}^{B} \mathcal{S}_i \right| + c_{\text{comp}} \cdot B \cdot k \tag{1}$$

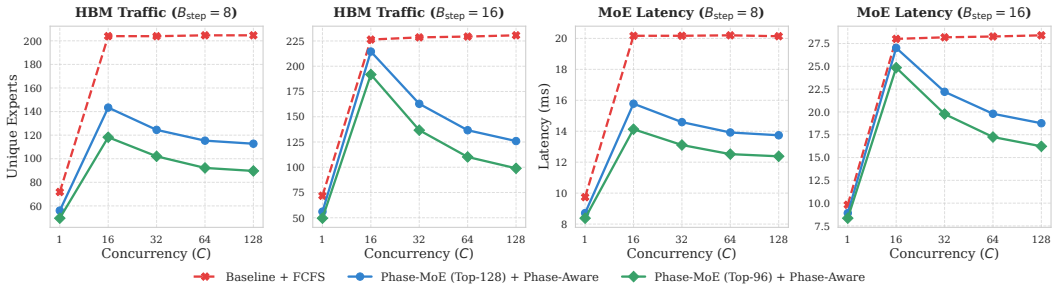


Figure 1: **System scalability and memory traffic under continuous batching.** Comparison of standard serving (*Baseline + FCFS*) against our co-design (*Phase-MoE + Phase-Aware*) across request concurrencies ($C \in \{1, 16, 32, 64, 128\}$) and step batch sizes ($B_{\text{step}} \in \{8, 16\}$). **Left:** Unique experts fetched from HBM per step (U). **Right:** Mean MoE kernel latency during iterative block decoding. While the baseline rapidly saturates the total expert pool ($U = \left|\bigcup_{m=1}^{B_{\text{step}}} \left(\bigcup_{i=1}^{B} \mathcal{S}_{m,i}\right)\right| \to N$, where $\mathcal{S}_{m,i}$ is the routed expert set for token i in sequence m), causing latency to hit the HBM bandwidth ceiling, Phase-MoE with Phase-Aware scheduling strictly bounds activations within phase candidate pools ($U \ll N$), unlocking high-concurrency scaling. (See Figure 7 for full results.)

where $\mathcal{S}_i \subset \{1, \ldots, N\}$ is the set of k experts routed for token i, c_{mem} is the fixed cost of fetching an expert's weights from HBM to SRAM, and c_{comp} is the marginal compute cost per token.

In autoregressive decoding ($B = 1$), the unique experts fetched are bounded ($|\bigcup \mathcal{S}_i| = k$). In BDLMs, processing B tokens simultaneously triggers an *Expert Explosion*. Because intra-block tokens are heterogeneous, the union of selected experts frequently expands toward pool capacity ($|\bigcup \mathcal{S}_i| \to N$). As highlighted by Taïga et al. (2026), diffusion MoEs can fetch an order of magnitude more experts per step than AR models, overwhelming compute savings with prohibitive HBM traffic.

The Heterogeneity of Continuous Batching. Recent interventions—such as sequence-level sharing (DES) (Chen et al., 2026), spatio-temporal reuse (TEAM) (Wei et al., 2026), or block-level aggregation (dMoE) (Feng et al., 2026)—mitigate this footprint in single-sequence decoding by exploiting the shared semantic context and uniform temporal state of a single sequence. However, production continuous batching (Yu et al., 2022; Kwon et al., 2023b) processes asynchronous requests simultaneously, introducing two compounding sources of heterogeneity:

- **Temporal Heterogeneity ($\Delta\varphi > 0$):** Asynchronous sequences reside at different denoising stages (block mask densities φ, defined in Section 3.1). Because discrete diffusion transitions from global planning (high φ) to local syntactic refinement (low φ), expert demands diverge across the batch.
- **Semantic Heterogeneity:** Concurrent prompts span disparate domains (e.g., multi-step reasoning vs. factual retrieval), activating disjoint expert subsets.

As shown in Figure 1, this compound heterogeneity drives standard serving (*Baseline + FCFS*) from ≈ 74 experts ($N = 256$) at $B_{\text{step}} = 1$ to > 225 at $B_{\text{step}} = 16$, hitting the HBM bandwidth ceiling.

The Analytical Dilemma. When addressing this memory bottleneck via post-hoc mitigations, managing multi-sequence heterogeneity introduces an inherent design trade-off:

1. **Allow Independence (Sacrifice Throughput):** Independent routing preserves fidelity, but batch-level expert demand explodes ($U = \left|\bigcup_{m=1}^{B_{\text{step}}} \bigcup_{i=1}^{B} \mathcal{S}_{m,i}\right| \to N$), saturating memory bandwidth.
2. **Force Batch Consensus (Sacrifice Representation):** Forcing a common routing intersection across an asynchronous batch bounds HBM memory traffic, but restricting disparate denoising phases and semantic domains to a shared coreset compromises specialized model capacity.

Consequently, resolving this memory expansion purely through post-hoc interventions remains challenging under continuous batching. This fundamental conflict necessitates a structural, training-time resolution (Section 3). As we empirically demonstrate in Section 5.3, by embedding these constraints into the architecture we can break this Pareto trade-off during inference.

3 TRAINING-TIME GUARANTEE: PHASE-CONSTRAINED MOE

To break this trade-off, Phase-Constrained MoE embeds support-sparsity bounds directly into the router during training via a phase-bias curriculum.

3.1 PHASE-LOCALIZED ROUTING AND PHASE 0 ISOLATION

Let $\varphi_i \in \{0, 1, \ldots, B\}$ be the temporal denoising phase of token i, defined by the number of remaining masked tokens in its block (which decreases non-unitarily as multiple tokens are unmasked in parallel). To structurally constrain expert activations, we extend auxiliary-loss-free load balancing (Wang et al., 2024) into a 2D constraint. We introduce a phase-bias matrix $M \in \mathbb{R}^{(B+1)\times N}$, explicitly conditioning the Top-k routing decision on the temporal phase:

$$\mathcal{S}_i = \underset{j\in\{1,\ldots,N\}}{\text{argTopK}} \left(s_{j,i} + M_{\varphi_i,j},\, k\right), \quad \text{where } s_{j,i} \text{ is the original gating score.} \tag{2}$$

A cornerstone of this formulation is *Phase 0 Isolation*. Drawing from AR inference analyses demonstrating a compute-memory dichotomy (Patel et al., 2024; Agrawal et al., 2024; 2023) (Appendix A.3.1), we partition block diffusion execution into two regimes. Phase 0 ($\varphi_i = 0$) governs fully revealed tokens during prefill or context commitment; operating over large contiguous matrices, it is *compute-bound*. Conversely, Phase $\varphi > 0$ governs iterative denoising; processing sparse masked tokens drastically drops arithmetic intensity, pushing execution into the *memory-bound* regime.

Consequently, we leave the prefill sub-matrix $M_{0,:}$ globally unconstrained. Simultaneously, the denoising sub-matrix $M_{>0,:}$ is strictly shaped and pruned to guarantee bounded support sparsity.

3.2 SOFT-TO-HARD OPTIMIZATION CURRICULUM

Enforcing strict phase-bounded routing immediately upon initialization causes catastrophic semantic collapse (as empirically demonstrated in Appendix G.5). We mitigate this via a two-stage curriculum that smoothly localizes expert utility before locking the architectural constraints. Let $L \in \mathbb{N}_0^{B\times N}$ be the aggregated token load over a micro-batch, where $L_{\varphi,j} = \sum_{i:\varphi_i=\varphi} \mathbb{I}(j \in \mathcal{S}_i)$ denotes the count of tokens at phase $\varphi \in \{1, \ldots, B\}$ routed to expert j.

Stage 1: Soft Shaping. We update $M_{>0}$ via a leaky integrator driven by two competing forces: a load-balancing push (λ_{lb}) to prevent expert saturation, and a phase-specialization pull (λ_{spec}) to attract experts to specific temporal phases (see Appendix G.8 for an ablation of λ values):

$$\Delta M_{\varphi,j} = -\lambda_{\text{lb}} \text{sgn}(L_{\varphi,j} - \bar{L}_{\varphi,\cdot}) + \lambda_{\text{spec}} \text{sgn}(L_{\varphi,j} - \bar{L}_{\cdot,j}) \tag{3}$$

where $\bar{L}_{\varphi,\cdot} = \frac{1}{N}\sum_{j=1}^{N} L_{\varphi,j}$ is the average token load across all experts at phase φ, and $\bar{L}_{\cdot,j} = \frac{1}{B}\sum_{\varphi=1}^{B} L_{\varphi,j}$ is the average load across all active denoising phases for expert j. To strongly bias contiguous denoising phases to share overlapping expert pools—a prerequisite for the Phase-Aware Scheduler to efficiently batch adjacent phases—we apply a 1D Gaussian blur ($\mathcal{G}_\sigma$) across the phase dimension independently for each expert j: $\tilde{M}_{>0,j} = (M_{>0,j} + \Delta M_{>0,j}) * \mathcal{G}_\sigma$, whose necessity is ablated in Appendix G.7. Finally, we apply masked double mean-centering to $\tilde{M}_{>0}$ to zero-center the router inputs, maintaining the shift-invariance of the gating function (refer to Appendix B for mathematical justification and Appendix G.6 for empirical validation). Intuitively, this combination of spatial blur and zero-sum centering enforces a natural specialization dynamic: an expert favoring a specific phase will smoothly extend its preference to adjacent phases while penalizing distant ones.

Stage 2: Hard Pruning. After sufficient phase-specialization, we enforce the hard capacity constraint $K_{\text{hard}} \ll N$. We truncate the continuous bias matrix to yield the strict support set $\mathcal{C}_\varphi$:

$$\mathcal{C}_\varphi = \underset{j\in\{1,\ldots,N\}}{\text{argTopK}}(M_{\varphi,j}, K_{\text{hard}}), \quad M_{\varphi,j} \leftarrow \begin{cases} M_{\varphi,j}, & j \in \mathcal{C}_\varphi \\ -\infty, & \text{otherwise} \end{cases} \quad (\forall \varphi \in \{1, \ldots, B\}) \tag{4}$$

Subsequent continual pre-training and downstream Supervised Fine-Tuning (SFT) proceed in Hard Mode. By restricting load-balancing updates to the active subset $\mathcal{C}_\varphi$, we cap HBM traffic per phase while preserving uniform expert utilization (see Figure 4 and the lifespan analysis in Section 5.1).

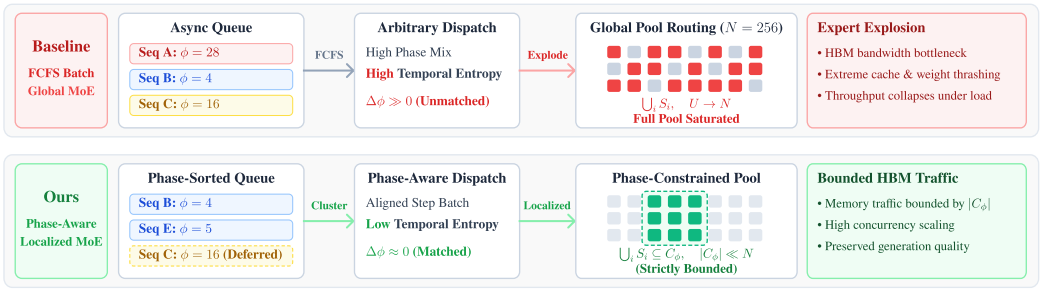


Figure 2: **System-level comparison of MoE continuous batching paradigms. Top (Baseline):** First-Come-First-Serve (FCFS) dispatch yields high temporal heterogeneity ($\Delta\varphi > 0$). Unconstrained routing ($N = 256$) causes an Expert Explosion where unique expert fetches approach the total pool capacity ($U \to N$), inducing HBM bandwidth bottlenecks and significant throughput degradation. **Bottom (Ours):** Phase-Aware Dispatch clusters sequences by mask density ($\Delta\varphi \approx 0$). By restricting routing to phase-constrained candidate pools $\mathcal{C}_\varphi$ ($|\mathcal{C}_\varphi| = K_{\text{hard}} \ll N$), unique expert activations are structurally bounded ($\bigcup \mathcal{S}_i \subseteq \mathcal{C}_\varphi$), preventing the Expert Explosion under continuous batching.

Algorithm 1 Phase-Aware Continuous Batching Scheduler

Require: Active request pool $\mathcal{P}$, execution batch size B_{step}, starvation threshold τ
$\mathcal{Q}_{\text{comp}}, \mathcal{Q}_{\text{dec}} \leftarrow$ Partition($\mathcal{P}$) ▷ Separate Phase $\varphi = 0$ (compute) and Phase $\varphi > 0$ (memory)
if $\mathcal{Q}_{\text{comp}} \neq \emptyset$ **then**
  return $\mathcal{Q}_{\text{comp}}[: B_{\text{step}}]$ ▷ Strict priority for compute-bound prefill/commit
end if
$\mathcal{S}_{\text{starved}} \leftarrow \{m \in \mathcal{Q}_{\text{dec}} \mid t_{\text{wait}}^{(m)} > \tau\}$
$m^* \leftarrow \arg\max_{m \in \mathcal{S}_{\text{starved}}} (t_{\text{wait}}^{(m)})$ if $\mathcal{S}_{\text{starved}} \neq \emptyset$ else null ▷ Identify the most starved request
Sort $\mathcal{Q}_{\text{dec}}$ in ascending order by mask density φ_m
$b \leftarrow \min(|\mathcal{Q}_{\text{dec}}|, B_{\text{step}})$ ▷ Adjust for under-filled queue
$\mathcal{W} \leftarrow \{W \subseteq \mathcal{Q}_{\text{dec}} \mid W \text{ is contiguous in } \mathcal{Q}_{\text{dec}}, |W| = b, \text{ and } (m^* \in W \text{ if } m^* \text{ exists})\}$
$\mathcal{E} \leftarrow \arg\min_{W \in \mathcal{W}} (\max_{m \in W} \varphi_m - \min_{m \in W} \varphi_m)$ ▷ Select batch minimizing phase spread $\Delta\varphi$
for $m \in \mathcal{Q}_{\text{dec}}$ **do** ▷ Update wait times
  $t_{m,\text{wait}} \leftarrow 0$ if $m \in \mathcal{E}$ else $t_{m,\text{wait}} + 1$
end for
return $\mathcal{E}$

4 Inference-Time Execution: Phase-Aware Scheduling

While Phase-Constrained MoE structurally limits the candidate expert pool per phase to $K_{\text{hard}} \ll N$, deploying these checkpoints under unconstrained continuous batching engines forfeits these memory-bandwidth guarantees. Because asynchronous requests arrive at arbitrary denoising stages, naive First-Come-First-Serve (FCFS) execution yields high temporal dispersion ($\Delta\varphi \gg 0$). The union of disparate phase pools across the batch quickly saturates the global expert capacity ($\bigcup_{m=1}^{B_{\text{step}}} \mathcal{C}_{\varphi_m} \to N$), recreating the memory-bandwidth bottleneck. We resolve this by pairing our architecture with a co-designed *Phase-Aware Continuous Batching Scheduler* (Algorithm 1, Figure 2).

4.1 Compute–Memory Disaggregation and Priority Queuing

Following the compute-memory dichotomy established in Section 3.1, the scheduling engine partitions active requests $\mathcal{P}$ into two specialized queues:

- **Queue 0 ($\mathcal{Q}_{\text{comp}}$, Phase $\varphi = 0$):** Contains sequences undergoing prompt prefilling or committing fully denoised blocks to the cache. Though *infrequent* relative to iterative decoding (see Appendix F.3), these forward passes are *compute-bound* and leverage unconstrained routing ($M_{0,:}$). $\mathcal{Q}_{\text{comp}}$ is prioritized to clear context boundaries, bounded by the concurrency limits.
- **Queue $\varphi > 0$ ($\mathcal{Q}_{\text{dec}}$, Phase $\varphi > 0$):** Contains sequences executing iterative denoising over masked block tokens. These passes are *memory-bound* and execute within phase-localized pools $\mathcal{C}_\varphi$.

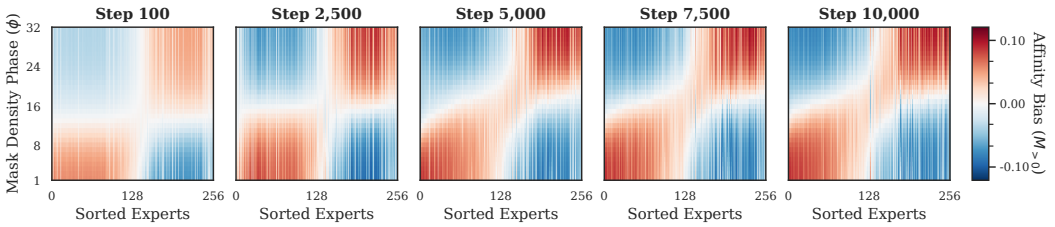


Figure 3: **Temporal evolution of Phase-MoE bias ($M_{>0}$) during Soft Shaping.** Heatmaps depict the 2D routing bias across training steps. Driven by the competing load-balancing push (λ_{lb}) and specialization pull (λ_{spec}) defined in Section 3.2, the router organically self-organizes into smooth, contiguous diagonal support bands across the denoising trajectory.

4.2 Sliding-Window Phase Alignment and Starvation Bounds

To schedule an execution step batch of size B_{step} from $\mathcal{Q}_{\text{dec}}$, the scheduler sorts all active decoding sequences by their current mask density φ_m. Because BDLMs apply a uniform mask density across the current decoding block, this sequence-level phase inherently defines the phase φ_i of all its constituent tokens. It then slides an evaluation window across $\mathcal{Q}_{\text{dec}}$ to identify the contiguous batch $\mathcal{E}$ that minimizes the intra-batch phase spread $\Delta\varphi = \max_{m\in\mathcal{E}} \varphi_m - \min_{m\in\mathcal{E}} \varphi_m$.

Because adjacent phase pools exhibit strong support overlap ($\mathcal{C}_\varphi \approx \mathcal{C}_{\varphi\pm1}$), minimizing $\Delta\varphi \to 0$ empirically constrains the batch-level expert union near $K_{\text{hard}} \ll N$ (formal analysis in Appendix B.3), curbing the Expert Explosion during continuous batching (Figure 1).

Starvation Mitigation. While processed sequences decrease in mask density ($\varphi \to 0$) and rapidly clear the decode queue, minority-phase requests can stall if bypassed by dense clusters. To mitigate unbounded worst-case wait times, we enforce a dynamic starvation threshold $\tau = \max(\tau_0, 2\lfloor|\mathcal{Q}_{\text{dec}}|/B_{\text{step}}\rfloor)$, where τ_0 is the minimum wait tolerance. When a request's queue age exceeds τ, the scheduler anchors the sliding window around this starved request ($m^* = \arg\max_m t_{\text{wait}}^{(m)}$) and packs the remaining $B_{\text{step}} - 1$ slots with sequences having minimal relative phase distance to m^*. This acts as a strict priority trigger to suppress tail starvation while preserving maximal support locality. For an empirical analysis of $\Delta\varphi$ scaling, scheduling overhead profiling, and tail-end starvation metrics (Time To Completion), refer to Appendix F.

5 Experiments

To rigorously evaluate Phase-Constrained MoE, we employ controlled A/B testing. Our objectives are twofold: first, to isolate the structural impact of phase-localized routing from the confounding effects of complex data pipelines prevalent in frontier models (Bie et al., 2025); and second, to evaluate how our co-design and existing baselines navigate the compound heterogeneity of continuous batching.

5.1 Experimental Setup

Exploiting the AR and BDLM equivalence at $B = 1$ (Taïga et al., 2026; Bie et al., 2025), we initialize all variants from a shared AR checkpoint to ensure uniform priors and eliminate pre-training variance.

Initialization & Curriculum: We branch from `Ling-mini-2.0` (Team et al., 2025) and use a controlled two-stage budget to isolate architectural impact: 10B tokens on SmolLM-Corpus (Ben Allal et al., 2024) for Continual Pre-Training (CPT), followed by 2B tokens on OpenHermes2.5 (Teknium, 2023) for Supervised Fine-Tuning (SFT). Figure 3 visualizes how Soft Shaping (Section 3.2) forms temporal phase affinities during early CPT (see Appendix C for hardware and hyperparameter details).

Evaluated Configurations ($B = 32$): We benchmark the unconstrained *Baseline* against two strictly bounded architectures: *Phase-MoE (Top-128)* and *Phase-MoE (Top-96)*. Our primary setting adopts $B = 32$; comprehensive ablations exploring parallel block size dynamics ($B \in \{16, 64\}$), Phase 0 isolation, extreme sparsity limits ($K_{\text{hard}} = 64$), and training dynamics are provided in Appendix G.

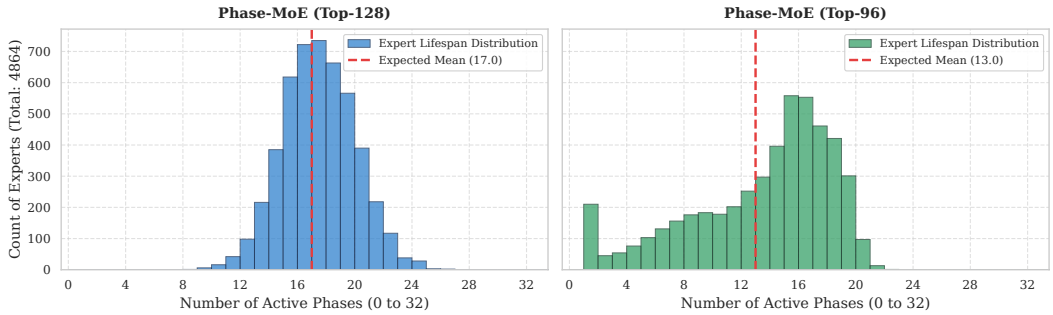


Figure 4: **Distribution of active expert lifespans across denoising phases.** Active phase count per expert ($N = 256$, across all 19 MoE layers, total 4,864 experts) under hard pruning for *Top-128* (*left*) and *Top-96* (*right*). Dashed red lines mark theoretical expected means ($1 + \frac{K_{\text{hard}}}{N} \cdot B$). Both configurations exhibit well-calibrated distributions centered near expectation without routing collapse.

Expert Lifespan and Routing Health. To verify that hard pruning avoids representation collapse, Figure 4 plots the distribution of expert phase lifespans ($|\{\varphi \mid j \in \mathcal{C}_\varphi\}|$) across all 19 MoE layers (4,864 experts total). Both Top-128 and Top-96 configurations exhibit well-behaved, unimodal distributions centered at their theoretical expected means ($1 + \frac{K_{\text{hard}}}{N} \cdot B = 17.0$ and 13.0 phases, respectively, including unconstrained Phase 0). Crucially, two properties hold:

- **No Generalist Omnipotence:** The distributions fall to zero long before the global limit, demonstrating that experts confine their support sets to distinct denoising phases.
- **Preserved Parametric Health:** No experts are dead. Although extreme sparsity limits in Top-96 drive a small cluster to operate solely within Phase 0, the load-balancing term (λ_{lb}) successfully prevents representation collapse, maintaining full active participation across the expert pool.

5.2 Generative Benchmark Performance

We evaluate downstream capabilities across five core domains: *Knowledge* (ARC-C/E (Clark et al., 2018), MMLU (Hendrycks et al., 2020)), *Reasoning* (HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020)), *Math* (GSM8K (Cobbe et al., 2021)), *Coding* (HumanEval (Chen et al., 2021)), and *Alignment* (IFEval (Zhou et al., 2023)). Results are summarized in Table 1.

Table 1: **Generative Benchmark Performance.** Comparison of the unconstrained baseline against Phase-MoE variants. The `llada2.0-mini-preview` checkpoint serves as an external reference.

	Knowledge			Reasoning		Math	Coding	Alignment
Model Configuration	**ARC-C**	**ARC-E**	**MMLU**	**He-Swag**	**PIQA**	**GSM8K**	**HumEval**	**IFEval**
`llada2.0-mini-preview`	0.892	–	0.725	0.740	0.843	0.890	0.805	0.625
Baseline ($B = 32$)	0.861	0.959	0.666	0.632	0.801	0.588	0.602	0.543
Phase-MoE (Top-128)	0.845	0.946	0.631	0.615	0.788	0.553	0.570	0.493
Phase-MoE (Top-96)	0.834	0.933	0.620	0.594	0.781	0.524	0.531	0.468

Transitioning from unconstrained routing to phase-localized pools introduces an expected performance trade-off. Phase-MoE (Top-128) yields slight drops across knowledge (ARC-C: $0.861 \rightarrow 0.845$) and reasoning (PIQA: $0.801 \rightarrow 0.788$). Restricting capacity further to Top-96 induces a steeper decline, particularly in mathematics and coding tasks.

Crucially, this empirical trade-off is a deliberate system-level compromise. While a performance gap exists between our controlled models and frontier checkpoints like `llada2.0-mini-preview`, our experimental setup is designed to maximize simplicity and experimental control. We intentionally omit massive data mixtures (> 100B tokens), RL alignment, and auxiliary confidence losses (detailed extensively in Appendix C.1). By stripping away these data-heavy enhancements, we isolate and prove that this minor accuracy drop directly buys a strict architectural bound on HBM traffic.

5.3 Comparison with Post-Hoc Routing Interventions

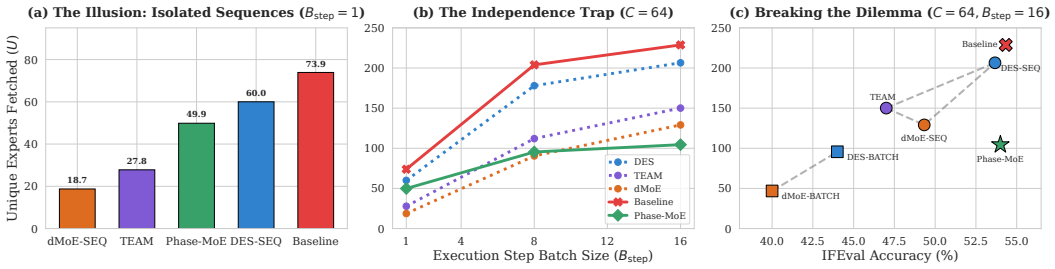


Figure 5: **The Continuous Batching Dilemma.** Evaluation of Phase-MoE against SOTA baselines (DES, TEAM, dMoE) under varying execution scales. **(a) The Illusion of Isolated Execution ($B_{\text{step}} = 1$):** Under single-sequence generation, other baselines successfully cull unique expert activations (U). **(b) The Independence Trap ($C = 64$):** When preserving sequence-level independence to maintain accuracy, prior methods fail to bound batch-level memory traffic. As B_{step} scales, the union of temporally and semantically disjoint expert sets saturates HBM bandwidth. **(c) Breaking the Dilemma ($C = 64, B_{\text{step}} = 16$):** Post-hoc methods face a trade-off under continuous batching. Forcing a batch-level consensus (DES-BATCH, dMoE-BATCH) caps memory traffic but triggers semantic collapse. Conversely, preserving independent routing (DES-SEQ, dMoE-SEQ, and TEAM; see Appendix E.5 for TEAM's batch-equivalence) prioritizes representational capacity but yields an Expert Explosion. Phase-MoE breaks this trade-off, leveraging its training-time structural bounds to minimize unique expert fetches with minimal degradation. (See Appendix E for full results)

In Section 5.2, we established that Phase-MoE's structural bounds require a minor compromise in isolated generative accuracy. We now demonstrate why this trade-off is justified for production serving. We compare Phase-MoE against three state-of-the-art post-hoc interventions: Dynamic Expert Sharing (DES, specifically DES-Vote) (Chen et al., 2026), TEAM (Wei et al., 2026), and dMoE (Feng et al., 2026). To evaluate their resilience to multi-sequence heterogeneity, we simulate production continuous batching (detailed in Appendix E) at sustained concurrency ($C = 64$) across semantically mixed workloads. Because continuous batching processes B_{step} asynchronous sequences, any post-hoc heuristic must choose an implementation paradigm: apply independently per sequence (*-SEQ*), or force a global consensus across the entire step batch (*-BATCH*).

The Single-Sequence Illusion. In isolation ($B_{\text{step}} = 1$), post-hoc mitigations—both inference heuristics (DES, TEAM) and distillation (dMoE)—exploit single-sequence temporal and semantic homogeneity to suppress expert activations ($U = 18.7$ for dMoE-SEQ, $U = 27.8$ for TEAM vs. Phase-MoE's $U = 49.9$; Figure 5a). However, this efficiency is pyrrhic: runtime coreset, multi-round, or top-p filtering introduces up to 9.1 ms of computational overhead. Crucially, even assuming custom fused kernels eliminate this tax, these isolated designs face a fundamental trade-off when scaled.

The Continuous Batching Dilemma. The compound semantic and temporal heterogeneity of continuous batching exposes the critical flaw of post-hoc methods. Because asynchronous sequences demand disjoint expert sets, these interventions are forced into a Pareto trade-off (Figure 5b, c):

- **The Independence Trap (-SEQ Variants):** Preserving independent sequence-level routing maintains generative fidelity but fails to bound batch-level memory traffic. As execution scales, the union of disjoint localized sets expands rapidly (Figure 5b). At $B_{\text{step}} = 16$, DES-SEQ nearly re-saturates the unconstrained baseline ($U = 206.5$ vs. 228.8), while dMoE-SEQ (129.1) and TEAM (150.1) scale steeply, both surpassing Phase-MoE's bound (104.5). *(TEAM is plotted singularly as its -SEQ and -BATCH bounds are mathematically identical; see Appendix E.5 for the formal proof.)*
- **The Consensus Trap (-BATCH Variants):** To forcibly cap HBM traffic, post-hoc mitigations can calculate a single, global consensus across the entire asynchronous batch. While this successfully bounds memory fetches (e.g., dMoE-BATCH limits U to 46.8), it triggers semantic collapse. Forcing temporally disjoint ($\Delta\varphi \gg 0$) and semantically unrelated sequences into an artificial routing bottleneck deprives them of essential representational capacity, causing IFEval accuracy to plummet to 40.0% for dMoE-BATCH and 44.0% for DES-BATCH.

Breaking the Dilemma. Phase-MoE resolves this bottleneck through its training-time architecture. Because the soft-to-hard curriculum aligns adjacent denoising phases to share expert support, the lightweight Phase-Aware scheduler (≤ 0.05 ms CPU overhead) can tightly cluster sequences without forcing an out-of-distribution consensus. This guaranteed synergy collapses the HBM bottleneck ($U = 104.5$ for Top-96 at $B_{\text{step}} = 16$) while preserving high baseline accuracy (54.0%), escaping the Pareto trade-off. Comprehensive hardware metrics and derivations are detailed in Appendix E.

5.4 Throughput & Continuous Batching Scalability

Having demonstrated that prior baselines break down under batching, we move beyond single-sequence execution (profiled in Appendix D) to evaluate system scalability across request concurrencies (C) and step batch sizes (B_{step}). Using the multi-domain workload established in Section 5.3 (detailed in Appendix E.1), we benchmark standard First-Come-First-Serve (FCFS) dispatch against our *Phase-Aware (PA)* scheduler, tracking unique expert memory fetches (U) and MoE kernel latency.

Table 2: **Continuous Batching Simulation Results** ($B_{\text{step}} = 8$, expert Pool $N = 256$). Comparison of Baseline and Phase-MoE variants across request concurrencies ($C \in \{16, 32, 64, 128\}$) under FCFS and Phase-Aware (PA) scheduling. *Experts* (U) is the average number of experts fetched from HBM per step, *Latency* is the mean MoE kernel time in milliseconds (See Table 7 for full results).

		$C = 16$		$C = 32$		$C = 64$		$C = 128$	
Model	**Scheduler**	**Experts**	**Latency**	**Experts**	**Latency**	**Experts**	**Latency**	**Experts**	**Latency**
Baseline	FCFS	203.98	20.17	203.92	20.17	204.67	20.20	204.67	20.14
	PA	194.59	19.44	190.81	19.25	187.76	19.02	187.65	19.07
Phase-MoE (Top-128)	FCFS	187.86	19.01	191.02	19.19	192.90	19.33	192.37	19.28
	PA	143.37	15.78	124.48	14.59	115.34	13.92	112.73	13.74
Phase-MoE (Top-96)	FCFS	168.81	17.55	172.01	17.85	173.02	17.79	172.65	17.78
	PA	**118.24**	**14.13**	**102.05**	**13.11**	**92.20**	**12.52**	**89.65**	**12.38**

Table 2 validates the dilemma formalized in Section 2: neither scheduling intervention nor phase-constrained routing resolves the bottleneck in isolation. Specifically, relying on scheduling alone (*Baseline + PA*) achieves near-perfect temporal alignment at $C = 128$ ($\Delta\varphi \approx 1.05$, see Appendix F), but unconstrained token routing still fetches 187.65 unique experts, stalling at 19.07 ms. Conversely, relying on phase candidate pools alone under naive dispatch (*Phase-MoE Top-96 + FCFS*) suffers from high temporal variance; the disjoint phase pools across asynchronous requests balloon the expert union back to 172.65, yielding a latency of 17.78 ms.

Eliminating this memory-bandwidth bottleneck requires the synergy of both techniques. Because Phase-MoE empirically promotes support overlap across adjacent denoising phases (Figure 6), the PA scheduler successfully curtails global HBM-to-SRAM transfers. Combined, *Phase-MoE (Top-96) + PA Scheduler* requires only 89.65 expert fetches, reducing kernel latency to 12.38 ms (a 38.5% improvement over the baseline) and effectively decoupling memory traffic from request concurrency.

6 Conclusion

Scaling Block Diffusion Language Models (BDLMs) via Mixture-of-Experts introduces a memory-bandwidth bottleneck: the *Expert Explosion*. Under the compound temporal and semantic heterogeneity of production continuous batching, post-hoc routing mitigations face challenging trade-offs between batch-level memory expansion and representational flexibility. We resolve this dilemma through a bandwidth-aware co-design that shifts the bottleneck resolution to an architectural guarantee. During training, Phase-Constrained MoE employs a soft-to-hard curriculum to structurally localize expert pools to denoising phases, leaving compute-bound prefill and block commitment unconstrained. During inference, the Phase-Aware Scheduler groups asynchronous requests by mask density to minimize intra-batch temporal spread. By maximizing overlapping expert support across concurrent sequences, this synergy decouples memory traffic from batch concurrency. Our framework cuts average expert activations by up to 56.2% and MoE kernel latency by up to 38.5%, while preserving competitive generative performance. Ultimately, this co-design establishes a scalable foundation for deploying MoE-based discrete diffusion models in high-throughput production environments.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the manuscript's writing and for assistance with code generation. We have reviewed all AI-assisted work, performing rigorous manual checks on all LLM-generated code and verifying the accuracy of all refined text. We take full responsibility for the content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken the following steps to ensure the reproducibility of this work:

- **Code:** The complete, anonymized source code for training (Phase-Constrained MoE) and inference (Phase-Aware Scheduler) is provided in the supplementary materials.
- **Theory and Proofs:** The formal derivation bounding the batch-level expert union and the mathematical justification for the Soft Shaping dynamics are fully detailed in Appendix B.
- **Algorithms:** The continuous batching dispatch logic is explicitly formulated in Algorithm 1.
- **Experimental Setup:** Appendix C details the hardware infrastructure, datasets, and complete hyperparameter grids (Table 3).
- **Ablations:** To ensure our findings isolate the architectural impact, comprehensive baselines and ablations isolating the routing constraints are provided in Appendix G.

REFERENCES

Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 117–134, 2024.

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021a.

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

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. SmolLM-Corpus: A curated collection of high-quality educational and synthetic data. https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus, July 2024. Hugging Face Dataset.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

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, Jiamei Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Junlin Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b. *ArXiv*, abs/2512.15745, 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Kun Cheng, Xiao He, Lei Yu, Zhijun Tu, Mingrui Zhu, Nannan Wang, Xinbo Gao, and Jie Hu. Diff-moe: Diffusion transformer with time-aware and space-adaptive experts. In *Forty-second International Conference on Machine Learning*, 2025a.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *ArXiv*, abs/2510.06303, 2025b.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

Nima Fathi, Torsten Scholak, and Pierre-André Noël. Unifying autoregressive and diffusion-based sequence generation. *arXiv preprint arXiv:2504.06416*, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Sicheng Feng, Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. dmoe: dllms with learnable block experts. *arXiv preprint arXiv:2605.30876*, 2026.

Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. Reverse training to nurse the reversal curse. *CoRR*, abs/2403.13799, 2024. doi: 10.48550/ARXIV.2403.13799.

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.

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, et al. Advancing expert specialization for better moe. *Advances in Neural Information Processing Systems*, 38:48767–48809, 2026.

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

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

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *CoRR*, abs/2507.18578, 2025. doi: 10.48550/ARXIV.2507.18578.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

InclusionAI. dFactory: Easy and Efficient dLLM Fine-Tuning. https://github.com/inclusionAI/dFactory, 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, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023a. doi: 10.1145/3600006.3613165.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023b.

Yunsung Lee, JinYoung Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13427–13436, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

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 preprint arXiv:2508.19982*, 2025.

Yahui Liu, Yang Yue, Jingyuan Zhang, Chenxi Sun, Yang Zhou, Wencong Zeng, Ruiming Tang, and Guorui Zhou. Efficient training of diffusion mixture-of-experts models: A practical recipe. *arXiv preprint arXiv:2512.01252*, 2025.

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

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. *CoRR*, abs/2509.26432, 2025. doi: 10.48550/ARXIV.2509.26432.

Zhicong Lu, Li Jin, Peiguang Li, Yu Tian, Linhao Zhang, Sirui Wang, Guangluan Xu, Changyuan Tian, and Xunliang Cai. Rethinking the reversal curse of llms: a prescription from human knowledge reversal. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 7518–7530. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.428.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. Autonomy-of-experts models. *arXiv preprint arXiv:2501.13074*, 2025.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *International Conference on Learning Representations*, volume 2026, pp. 75251–75271, 2026.

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.

Taishi Nakamura, Satoki Ishikawa, Masaki Kawamura, Okamoto Okamoto, Daisuke Nohara, Jun Suzuki, and Rio Yokota. Optimal sparsity of mixture-of-experts language models for reasoning tasks. In *International Conference on Learning Representations*, volume 2026, pp. 104830–104854, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Jirong Wen, and Chongxuan Li. Large language diffusion models. *ArXiv*, abs/2502.09992, 2025.

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.

Byeongjun Park, Hyojun Go, Jin-Young Kim, Sangmin Woo, Seokil Ham, and Changick Kim. Switch diffusion transformer: Synergizing denoising tasks with sparse mixture-of-experts. In *European Conference on Computer Vision*, pp. 461–477. Springer, 2024a.

Byeongjun Park, Sangmin Woo, Hyojun Go, Jin-Young Kim, and Changick Kim. Denoising task routing for diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 10334–10358, 2024b.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In Dawn Song, Michael Carbin, and Tianqi Chen (eds.), *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org, 2023.

Xiaojing Qi, Lun Du, Xinyuan Zhang, Lanning Wei, Tao Jin, and Da Zheng. Hierarchy decoding: A training-free parallel decoding strategy for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *ArXiv*, abs/2406.07524, 2024b.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, et al. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.

Haotian Sun, Tao Lei, Bowen Zhang, Yanghao Li, Haoshuo Huang, Ruoming Pang, Bo Dai, and Nan Du. Ec-dit: Scaling diffusion transformers with adaptive expert-choice routing. In *International Conference on Learning Representations*, volume 2025, pp. 72383–72401, 2025.

Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, Tianlin Liu, Ivan Lobov, Ofir Nabati, João Gabriel Oliveira, Nicolas Perez-Nieves, Nastasia Prutianova, Bobak Shahriari, Jean Tarbouriech, Pavel Tyletski, Çağlar Ünlü, Cindy Wu, Glenn Cameron, Jerome T. Connor, Sertan Girgin, Maarten R. Grootendorst, Alon Levkovitch, Eliya Nachmani, Omar Sanseviero, Piotr Stańczyk, Quentin Berthet, Andrew Campbell, Clément Crepy, Valentin De Bortoli, Arnaud Doucet, Romuald Élie, Alexandre Galashov, Klaus Greff, Alexis Jacq, David Ruhe, Yu-Han Wu, Sebastian Flennerhag, Brendan O'Donoghue, George Scrivener, and Shantanu Thakoor. DiffusionGemma technical report. *arXiv preprint*, 2026.

Ling Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation. *arXiv preprint arXiv:2510.22115*, 2025.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Boshi Wang and Huan Sun. Is the reversal curse a binding problem? uncovering limitations of transformers from a basic generalization failure. *CoRR*, abs/2504.01928, 2025. doi: 10.48550/ARXIV.2504.01928.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. Time is a feature: Exploiting temporal dynamics in diffusion language models. In *International Conference on Learning Representations*, volume 2026, pp. 72909–72935, 2026.

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.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

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

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. *CoRR*, abs/2505.22618, 2025b. doi: 10.48550/ARXIV.2505.22618.

Tianyi Wu, Xiaoxi Sun, Yanhua Jiao, Yulin Li, Yixin Chen, Yue Cao, YiQi Hu, and Zhuotao Tian. Dynamic-dllm: Dynamic cache-budget and adaptive parallel decoding for training-free acceleration of diffusion LLM. *CoRR*, abs/2606.26120, 2026. doi: 10.48550/ARXIV.2606.26120.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX symposium on operating systems design and implementation (OSDI 22)*, pp. 521–538, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

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

Jinze Zhao, Peihao Wang, Junjie Yang, Ruisi Cai, Gaowen Liu, Jayanth Srinivasa, Ramana Rao Kompella, Yingbin Liang, and Zhangyang Wang. Sparse mixture-of-experts for compositional generalization: Empirical evidence and theoretical foundations of optimal sparsity. *arXiv preprint arXiv:2410.13964*, 2024.

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

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.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

TABLE OF NOTATION

Symbol	Description
Architecture & Diffusion Dynamics	
B	Parallel block size (number of tokens predicted simultaneously)
$\varphi \in \{0, 1, \dots, B\}$	Discrete mask density (temporal denoising phase) of a block
$\Delta\varphi$	Intra-batch phase spread ($\max_{m\in\mathcal{E}} \varphi_m - \min_{m\in\mathcal{E}} \varphi_m$)
N	Total number of routed experts in the MoE layer
k	Top-k expert routing capacity per token
K_{hard}	Maximum candidate expert pool size active per phase after hard pruning ($K_{\text{hard}} \ll N$)
MoE Routing & Optimization	
$M \in \mathbb{R}^{(B+1)\times N}$	2D phase-bias matrix conditioning routing on the temporal phase
$M_{0,:}, M_{>0,:}$	Bias sub-matrices for unconstrained prefill ($\varphi = 0$) and active denoising ($\varphi > 0$)
$s_{j,i}$	Unbiased gating score for expert j on token i
$\mathcal{S}_i$ (or $\mathcal{S}_{m,i}$)	Set of k routed experts for token i (in sequence m)
$\mathcal{C}_\varphi$	Strict candidate expert support set for phase φ ($\lvert\mathcal{C}_\varphi\rvert = K_{\text{hard}}$)
$J(A, B)$	Jaccard similarity index between sets A and B, defined as $\frac{\lvert A\cap B\rvert}{\lvert A\cup B\rvert}$
$L_{\varphi,j}$	Aggregated token load for expert j at phase φ over a micro-batch
$\lambda_{\text{lb}}, \lambda_{\text{spec}}$	Coefficients for load-balancing push and phase-specialization pull
$\mathcal{G}_\sigma$	1D Gaussian smoothing kernel with bandwidth σ
Systems & Continuous Batching	
B_{step}	Step batch size (number of concurrent sequences dispatched per forward pass)
C	Active request concurrency in the serving engine
U	Mean unique expert weights fetched from HBM to SRAM per forward step
T_{MoE}	MoE layer inference execution latency
$c_{\text{mem}}, c_{\text{comp}}$	Cost coefficients for HBM expert weight transfer and token arithmetic compute
$\mathcal{P}$	Global pool of active requests
$\mathcal{Q}_{\text{comp}}, \mathcal{Q}_{\text{dec}}$	Execution queues for compute-bound ($\varphi = 0$) and memory-bound ($\varphi > 0$) requests
$\mathcal{E}$	Scheduled contiguous execution batch minimizing phase spread
$t_{m,\text{wait}}$	Accumulated queue wait time (in steps) for sequence m
τ	Dynamic starvation threshold triggering worst-case priority scheduling

Table of Contents & Appendix

CONTENTS

1 Introduction **1**

2 Problem Formulation: The Continuous Batching Dilemma **2**

3 Training-Time Guarantee: Phase-Constrained MoE **4**

3.1 Phase-Localized Routing and Phase 0 Isolation 4

3.2 Soft-to-Hard Optimization Curriculum 4

4 Inference-Time Execution: Phase-Aware Scheduling **5**

4.1 Compute–Memory Disaggregation and Priority Queuing 5

4.2 Sliding-Window Phase Alignment and Starvation Bounds 6

5 Experiments **6**

5.1 Experimental Setup 6

5.2 Generative Benchmark Performance 7

5.3 Comparison with Post-Hoc Routing Interventions 8

5.4 Throughput & Continuous Batching Scalability 9

6 Conclusion **9**

A Background and Extended Related Work **19**

A.1 Autoregressive Language Modeling 19

A.2 Discrete Denoising Diffusion Processes 19

A.2.1 Forward Corruption Process 19

A.2.2 Reverse Denoising Process and Objective 20

A.3 Block Diffusion Language Models (BDLMs) 20

A.3.1 Temporal Denoising Phase and Mask Density 21

A.4 Mixture-of-Experts (MoE) and Arithmetic Constraints 21

A.4.1 Standard Top-k Routing 21

A.4.2 The Hardware Cost Model and the Expert Explosion 22

B Mathematical Justification for Phase-MoE Dynamics **22**

B.1 Gaussian Smoothing on the Absolute State 22

B.2 Masked Double Mean-Centering and Routing Domination 23

B.3 Formal Derivation of the Batch-Level Expert Bound 23

C Experimental Setup, Hardware, and A/B Testing Methodology **24**

C.1 A/B Testing Justification and the Performance Gap 24
C.2 Hardware Infrastructure 24
C.3 Hyperparameter Configurations and Gaussian Scaling 25

D Additional Empirical Results for Section 5 **25**
D.1 Single-Sequence Hardware Efficiency 25
D.2 Temporal Specialization Dynamics 26

E Comprehensive Evaluation of Prior Baselines under Continuous Batching **26**
E.1 Experimental Setup: Mixed Workloads and "Ghost Sequences" 26
E.2 Integration of Prior Routing Interventions 29
E.3 Performance in Isolated Execution ($B_{\text{step}} = 1$) 29
E.4 The Continuous Batching Dilemma ($B_{\text{step}} > 1$) 30
E.5 The Mathematical Tautology of TEAM's Hot-Cold Routing Constraint 31
E.6 Computational Overhead and the Custom Kernel Context 31

F Comprehensive Continuous Batching Results **31**
F.1 Scaling Dynamics and Memory Traffic 33
F.2 Phase Spread Alignment Dynamics ($\Delta\varphi$) 33
F.3 Compute–Memory Queue Dynamics: Phase 0 vs. Phase > 0 35
F.4 Starvation, Tail Latency, and Completion Dynamics 35
F.5 Scheduling Computational Overhead 35

G Ablation Studies and Analysis **36**
G.1 Metrics Formulation 36
G.2 Impact of Parallel Block Size (B) and Phase Granularity 37
G.3 Impact of Phase 0 Isolation (Prefill Constraints) 37
G.4 Impact of Pruning Aggressiveness (Support Sparsity Lower Bound) 38
G.5 Soft-to-Hard Curriculum vs. Static Initialization 38
G.6 Routing Dynamics: Mean-Centering and Routing Domination 38
G.7 Routing Dynamics: Gaussian Blur and Support Overlap 38
G.8 Push vs. Pull: Load Balancing and Specialization Trade-offs 39

H Broader Impacts **39**

A Background and Extended Related Work

This section establishes the foundational concepts necessary for understanding the architectures and algorithmic interventions proposed in this work. We begin by reviewing the traditional autoregressive paradigm and formally defining its intrinsic computational bottlenecks during inference. We then introduce discrete diffusion processes, which provide the theoretical substrate for parallel sequence generation. Building on these two paradigms, we formalize Block Diffusion Language Models (BDLMs), which structurally interpolate between causal and bidirectional modeling. Finally, we detail the mathematical formulation of Mixture-of-Experts (MoE) routing, culminating in a formal cost model of hardware execution that precisely defines the *Expert Explosion* memory-bandwidth bottleneck addressed in this work.

A.1 Autoregressive Language Modeling

Autoregressive (AR) sequence generation constitutes the dominant paradigm for modern large language models. Let $x = (x_1, \ldots, x_L)$ represent a sequence of L discrete tokens drawn from a finite vocabulary $\mathcal{V}$. Autoregressive models define a joint probability distribution over the sequence via the chain rule of probability, factorizing strictly from left to right:

$$p_\theta(x) = \prod_{i=1}^{L} p_\theta(x_i \mid x_{<i}) \tag{5}$$

where $x_{<i} = (x_1, \ldots, x_{i-1})$ denotes the causal context, and θ represents the learnable parameters of a neural network, typically parameterized by a Transformer architecture (Vaswani et al., 2017). The model parameters are optimized via maximum likelihood estimation (MLE), minimizing the negative log-likelihood over a given data distribution p_{data}:

$$\mathcal{L}_{\text{AR}}(\theta) = -\mathbb{E}_{x \sim p_{\text{data}}} \left[\sum_{i=1}^{L} \log p_\theta(x_i \mid x_{<i}) \right] \tag{6}$$

While the training phase can be efficiently parallelized across the sequence dimension, autoregressive inference is inherently sequential. Generating a sequence of length L fundamentally requires $\mathcal{O}(L)$ sequential forward passes. To circumvent the redundant $\mathcal{O}(i^2)$ recomputation of historical context representations for $x_{<i}$ at each step i, modern inference engines cache the intermediate Key and Value (KV) tensors of previously generated tokens (Pope et al., 2023; Kwon et al., 2023a).

Although KV-caching resolves the arithmetic redundancy, it transforms the generation process into a *memory-bound* operation. For every single token generated, the hardware accelerator must transfer the entire model weight matrices and the dynamically growing KV-cache (which scales as $\mathcal{O}(i)$ per sequence) from High-Bandwidth Memory (HBM) to the SRAM. Consequently, the arithmetic logic units (ALUs) remain underutilized while waiting for memory transfers, imposing a physical speed limit on AR decoding and motivating the exploration of parallel generation paradigms.

A.2 Discrete Denoising Diffusion Processes

To overcome the sequential bottleneck of AR modeling, diffusion models recast generation as a parallel, iterative denoising process. While originally developed for continuous Euclidean spaces (Ho et al., 2020), the framework has been formally extended to discrete categorical spaces, providing a principled method for modeling text and other discrete structures (Austin et al., 2021b).

A.2.1 Forward Corruption Process

Let $x^{(0)} \in \mathcal{V}^L$ denote a clean data sequence. A discrete diffusion model defines a forward Markov process that progressively corrupts $x^{(0)}$ over a sequence of discrete temporal states $x^{(1)}, \ldots, x^{(T)}$ (where T is the maximum diffusion timestep), approaching a known, structureless prior noise distribution p_{noise}. The transition at diffusion step t is governed by a categorical distribution applied independently to each token $i \in \{1, \ldots, L\}$:

$$q(x^{(t)} \mid x^{(t-1)}) = \prod_{i=1}^{L} \text{Cat}(x_i^{(t)} \mid Q_t \mathbf{e}_{x_i^{(t-1)}}) \tag{7}$$

where $Q_t \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a column-stochastic transition matrix defining the probability of transitioning between token states, and $\mathbf{e}_v$ is the one-hot encoding column vector of token v.

In text-based discrete diffusion, the *absorbing state* (or masking) transition is the most prevalent paradigm (Sahoo et al., 2024a). The base vocabulary is augmented with a special `[MASK]` token, yielding an extended state space $\mathcal{V}^+ = \mathcal{V} \cup \{\texttt{[MASK]}\}$. The transition matrix $Q_t \in \mathbb{R}^{|\mathcal{V}^+|\times|\mathcal{V}^+|}$ is parameterized such that true tokens are progressively replaced by `[MASK]` with a monotonically increasing probability, while masked tokens remain permanently masked (absorbed):

$$Q_t = \alpha_{t|t-1} I + (1 - \alpha_{t|t-1})\mathbf{m}\mathbf{1}^\top \tag{8}$$

where $\alpha_{t|t-1} \in [0, 1]$ controls the signal retention rate at step t, $\mathbf{m} \in \{0, 1\}^{|\mathcal{V}^+|}$ is the one-hot indicator vector for the `[MASK]` token, and $\mathbf{1} \in \mathbb{R}^{|\mathcal{V}^+|}$ is a vector of ones. Because the transition is Markovian and linearly defined, the multi-step marginal distribution $q(x^{(t)} \mid x^{(0)})$ can be computed analytically in closed form:

$$q(x^{(t)} \mid x^{(0)}) = \prod_{i=1}^{L} \mathrm{Cat}(x_i^{(t)} \mid \bar{\alpha}_t \mathbf{e}_{x_i^{(0)}} + (1 - \bar{\alpha}_t)\mathbf{m}), \quad \text{where } \bar{\alpha}_t = \prod_{\tau=1}^{t} \alpha_{\tau|\tau-1}. \tag{9}$$

A.2.2 Reverse Denoising Process and Objective

Sequence generation is achieved by simulating the reverse process $q(x^{(t-1)} \mid x^{(t)})$. By Bayes' theorem, the true reverse transition is $q(x^{(t-1)} \mid x^{(t)}) \propto q(x^{(t)} \mid x^{(t-1)})q(x^{(t-1)})$. Because the marginal $q(x^{(t-1)})$ depends on the intractable data distribution p_{data}, the exact reverse process is intractable and must be approximated by a parameterized neural network $p_\theta(x^{(t-1)} \mid x^{(t)})$. Assuming conditional independence across tokens given the fully noised sequence $x^{(t)}$, the generative trajectory is factorized as:

$$p_\theta(x^{(0:T)}) = p_{\text{noise}}(x^{(T)}) \prod_{t=1}^{T} \prod_{i=1}^{L} p_\theta(x_i^{(t-1)} \mid x^{(t)}) \tag{10}$$

The network parameters θ are optimized by maximizing the Evidence Lower Bound (ELBO) on the marginal likelihood. This process minimizes the Kullback-Leibler (KL) divergence between the true forward posterior $q(x^{(t-1)} \mid x^{(t)}, x^{(0)})$ and the parameterized reverse process $p_\theta(x^{(t-1)} \mid x^{(t)})$.

For masked diffusion processes, assuming the network directly predicts the clean data distribution $p_\theta(x^{(0)} \mid x^{(t)})$, the ELBO simplifies to a time-weighted cross-entropy loss evaluated exclusively over the currently masked positions (Shi et al., 2024):

$$\mathcal{L}_{\text{MDLM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(1,T),x^{(0)}\sim p_{\text{data}},x^{(t)}\sim q(\cdot|x^{(0)})} \left[\sum_{i:x_i^{(t)}=\texttt{[MASK]}} \omega(t) \log p_\theta(x_i^{(0)} \mid x^{(t)}) \right] \tag{11}$$

where $\omega(t)$ is a schedule-dependent continuous weighting function. Unlike causal AR models, this bidirectional formulation allows the model to leverage full-sequence, non-causal context to resolve dependencies, unlocking parallel decoding capabilities.

A.3 Block Diffusion Language Models (BDLMs)

Block Diffusion Language Models (BDLMs) (Arriola et al., 2025; Fathi et al., 2025) provide a structural interpolation between the strict causality of AR models and the bidirectional parallelization of pure diffusion models.

Instead of generating one token at a time, or alternatively denoising an entire full-length sequence simultaneously, a BDLM partitions a sequence of length L into contiguous blocks of size B. Sequence generation proceeds semi-autoregressively across these blocks:

$$\log p_\theta(x) = \sum_{b=1}^{\lceil L/B \rceil} \log p_\theta(\mathbf{x}_b \mid \mathbf{x}_{<b}) \tag{12}$$

where $\mathbf{x}_b = (x_{(b-1)B+1}, \ldots, x_{\min(bB,L)})$ represents the b-th generation block (with the final block truncated if L is not an exact multiple of B), and $\mathbf{x}_{<b}$ encapsulates all previously generated, fully denoised tokens.

Crucially, *inter-block* generation remains strictly causal. This preserves the ability to utilize the same efficient KV-caching mechanisms employed in standard AR models for the committed context $\mathbf{x}_{<b}$. Conversely, *intra-block* generation employs discrete diffusion. Within a target block b, the diffusion process refines the state $\mathbf{x}_b^{(t)}$ over multiple forward passes, explicitly exploiting the local bidirectional context among the B tokens.

A.3.1 Temporal Denoising Phase and Mask Density

As the discrete diffusion process refines a target block, the number of masked tokens progressively decreases. We formally define the *temporal denoising phase* (or mask density) of a block as $\varphi \in \{0, 1, \ldots, B\}$, representing the exact number of tokens within the decoding block that currently remain masked.

This formulation yields two distinct execution regimes from a hardware perspective:

1. **Active Denoising ($\varphi > 0$):** Represents the iterative, multi-step refinement of the block. Because the context $\mathbf{x}_{<b}$ is already committed and cached, the forward passes over the highly sparse subset of unmasked tokens within $\mathbf{x}_b$ exhibit low arithmetic intensity. Consequently, execution falls into the *memory-bound* regime.
2. **Context Commitment ($\varphi = 0$):** Represents the fully denoised block state. Once all B tokens are revealed, they are processed in a single, contiguous forward pass to update the global KV-cache for the subsequent block. This dense matrix multiplication across all B tokens operates efficiently in the *compute-bound* regime.

Because multiple tokens within the block may be confidently unmasked simultaneously in a single forward pass, the temporal phase φ decreases non-unitarily. This predictable trajectory—from global semantic planning (high φ) to local syntactic refinement (low φ)—constitutes the foundational temporal mechanic exploited by the *Phase-Constrained MoE* architecture introduced in Section 3.

A.4 Mixture-of-Experts (MoE) and Arithmetic Constraints

To scale the representational capacity of a neural network without inducing a proportional, linear increase in compute, the Mixture-of-Experts (MoE) architecture replaces dense feed-forward layers with a set of N structurally identical but parametrically distinct sub-networks, or "experts" (Shazeer et al., 2017; Fedus et al., 2022).

A.4.1 Standard Top-k Routing

For a given token i with a hidden representation $h_i \in \mathbb{R}^d$, a parameterized router network yields a probability distribution over the N experts. This is typically achieved via a linear projection $W_g \in \mathbb{R}^{N \times d}$ followed by a Softmax activation to produce normalized gating scores $s_{j,i} \in [0, 1]$ for each expert j. To guarantee sparsity, the router selects a subset of k experts ($k \ll N$), denoted $\mathcal{S}_i \subset \{1, \ldots, N\}$, defined formally as:

$$\mathcal{S}_i = \underset{j \in \{1,\ldots,N\}}{\text{argTopK}}(s_{j,i},\, k) \tag{13}$$

The final representation is computed as the linearly weighted combination of the outputs from the selected experts, renormalized over the active set $\mathcal{S}_i$:

$$\text{MoE}(h_i) = \sum_{j \in \mathcal{S}_i} \frac{s_{j,i}}{\sum_{l \in \mathcal{S}_i} s_{l,i}} E_j(h_i) \tag{14}$$

where $E_j(h_i)$ denotes the forward pass computation of the j-th expert.

A.4.2 The Hardware Cost Model and the Expert Explosion

The execution latency of an MoE layer, denoted T_{MoE}, is constrained by two physical hardware limits: arithmetic computation (measured in FLOPS) and memory bandwidth (the transfer of parameters from HBM to SRAM).

When processing a block of B tokens simultaneously, the total computational cost scales linearly with respect to the number of tokens B and the routing capacity k. However, the memory cost is determined not by the average routing capacity, but by the *union* of all unique experts activated across the entire sequence block. Chen et al. (2026); Oncescu et al. (2025) formalize this MoE layer execution latency as:

$$T_{\text{MoE}} \approx c_{\text{mem}} \cdot \left| \bigcup_{i=1}^{B} \mathcal{S}_i \right| + c_{\text{comp}} \cdot B \cdot k \tag{15}$$

where c_{mem} is the fixed latency cost of fetching a single expert's weight matrices from HBM, and c_{comp} is the marginal arithmetic computation cost per token.

Equation 15 exposes a critical systems-level vulnerability when MoE is combined with Block Diffusion. During standard autoregressive decoding ($B = 1$), the memory footprint is tightly bounded to the individual token's routing: $|\mathcal{S}_i| = k$. However, under BDLM parallel block decoding ($B \gg 1$), the union term becomes highly volatile. Because the B tokens unmasked simultaneously often represent highly semantic concepts, the router independently dispatches them to vastly different experts.

This intra-block semantic dispersion rapidly inflates the batch-level expert union, pushing $\left|\bigcup_{i=1}^{B} \mathcal{S}_i\right| \to N$. We define this acute degradation of sparsity as the *Expert Explosion*. Rather than fully utilizing the ALUs for parallel arithmetic, the hardware stalls while transferring massive amounts of expert parameters from HBM. As formalized in Section 2, when this intra-block semantic heterogeneity is compounded by the inter-sequence temporal heterogeneity inherent to production *continuous batching* systems (Yu et al., 2022; Kwon et al., 2023b), the foundational assumptions underlying post-hoc routing mitigations break down, necessitating the structural, training-time resolution proposed in this work.

B Mathematical Justification for Phase-MoE Dynamics

This section details the theoretical motivations behind the update operations applied during the Soft Shaping stage (Section 3.2).

B.1 Gaussian Smoothing on the Absolute State

During Soft Shaping, we apply a 1D Gaussian smoothing kernel $\mathcal{G}_\sigma$ across the phase dimension of the *absolute* bias matrix rather than solely to the delta updates. If smoothing were applied only to the updates (i.e., $M^{(t+1)} = M^{(t)} + \Delta M^{(t)} * \mathcal{G}_\sigma$), the absolute bias values M could grow unboundedly over time because the momentum integrates without decay.

Applying the convolution to the absolute state ($M^{(t+1)} = (M^{(t)} + \Delta M^{(t)}) * \mathcal{G}_\sigma$, where $*$ denotes column-wise 1D convolution) acts as a spatial low-pass filter. Because the Gaussian kernel integrates to 1, it strictly attenuates high-frequency spatial variations. When paired with the subsequent zero-mean centering, this dynamic mathematically prevents unbounded magnitude growth. It ensures the model adapts dynamically to recent data distributions rather than becoming anchored by historical inertia. Furthermore, this structural locality provides a strong empirical prior that adjacent phases (φ and $\varphi \pm 1$) converge toward highly overlapping expert support sets. It is crucial to note that continuous smoothing of the bias matrix does not strictly guarantee intersecting Top-k subsets, as the discrete argTopK operation is non-linear and sensitive to input-dependent gating volatility. Nonetheless, this spatial low-pass filter empirically maximizes the probability of support overlap, which is computationally vital for the Phase-Aware Scheduler.

B.2 Masked Double Mean-Centering and Routing Domination

The standard MoE routing decision utilizes a Softmax or Sigmoid gating function to produce normalized routing scores $s_{j,i} \in [0, 1]$. Because the phase-bias $M_{\varphi,j}$ is applied *post-activation* to these scores during the Top-k selection ($s_{j,i} + M_{\varphi,j}$), the magnitude of M directly competes with the model's dynamic gating outputs.

If a global drift in the phase-bias matrix M occurs—such that a specific phase φ accumulates a high average bias—the absolute magnitude of M could grow to overwhelm the bounded $[0, 1]$ range of the gating scores. This global shift would cause the static bias to completely dictate the Top-k ranking, rendering the input-dependent router scores irrelevant and effectively freezing the router's gradient-based learning.

To prevent this routing domination, the denoising bias sub-matrix $M_{>0}$ must strictly maintain a zero-mean property across both dimensions. Let $\tilde{M}_{>0} \in \mathbb{R}^{B\times N}$ be the bias matrix immediately after Gaussian smoothing. We compute the phase-wise mean, expert-wise mean, and global mean for each element:

$$\mu_{\text{row},\varphi} = \frac{1}{N}\sum_{j=1}^{N} \tilde{M}_{\varphi,j} \tag{16}$$

$$\mu_{\text{col},j} = \frac{1}{B}\sum_{\varphi=1}^{B} \tilde{M}_{\varphi,j} \tag{17}$$

$$\mu_{\text{global}} = \frac{1}{B \cdot N}\sum_{\varphi=1}^{B}\sum_{j=1}^{N} \tilde{M}_{\varphi,j} \tag{18}$$

The masked double mean-centered matrix element is computed as:

$$M_{\varphi,j} = \tilde{M}_{\varphi,j} - \mu_{\text{row},\varphi} - \mu_{\text{col},j} + \mu_{\text{global}} \tag{19}$$

This operation mathematically guarantees that $\sum_j M_{\varphi,j} = 0$ for all phases φ, and $\sum_\varphi M_{\varphi,j} = 0$ for all experts j. Consequently, the router purely adjusts *relative* expert preferences (specialization) without injecting a global magnitude shift that interferes with the original gating distribution scale.

B.3 Formal Derivation of the Batch-Level Expert Bound

We formalize the High-Bandwidth Memory (HBM) bound achieved by the synergy of Phase-Constrained MoE and the Phase-Aware Scheduler.

Proposition 1. *Given a continuous execution batch $\mathcal{E}$ of size B_{step}, where each sequence $m \in \mathcal{E}$ is at denoising phase φ_m, the unique expert memory traffic U is strictly decoupled from the batch size and bounded by K_{hard} in the limit of perfect phase alignment ($\Delta\varphi = 0$).*

Proof. Let $\mathcal{S}_{m,i}$ denote the set of k experts routed for token $i \in \{1, \dots, B\}$ in sequence m. The total unique experts fetched from HBM is the cardinality of the global batch union: $U = \left|\bigcup_{m\in\mathcal{E}} \bigcup_{i=1}^{B} \mathcal{S}_{m,i}\right|$.

Our Phase-Constrained MoE architecture (Section 3.2) structurally bounds the routed experts for any token to its phase-specific candidate pool: $\mathcal{S}_{m,i} \subseteq \mathcal{C}_{\varphi_m}$, with $|\mathcal{C}_{\varphi_m}| = K_{\text{hard}}$. Consequently, the batch-level traffic is bounded by the union of these active candidate pools:

$$U \le \left|\bigcup_{m\in\mathcal{E}} \mathcal{C}_{\varphi_m}\right| \tag{20}$$

During inference, the Phase-Aware Scheduler (Algorithm 1) actively groups sequences to minimize the intra-batch phase spread $\Delta\varphi = \max_{m\in\mathcal{E}} \varphi_m - \min_{m\in\mathcal{E}} \varphi_m$. In the limit of perfect alignment ($\Delta\varphi = 0$), all sequences reside at a singular phase $\bar{\varphi}$, causing the candidate pools to collapse into a single shared set ($\bigcup_{m\in\mathcal{E}} \mathcal{C}_{\varphi_m} = \mathcal{C}_{\bar{\varphi}}$). This yields the strict analytic bound:

$$U \le |\mathcal{C}_{\bar{\varphi}}| = K_{\text{hard}} \ll N \tag{21}$$

Furthermore, even when $\Delta\varphi > 0$ but small, the continuous 1D Gaussian smoothing regularizes the magnitude of support shift. While the discrete $\mathrm{argTopK}$ operation precludes a strict mathematical guarantee of overlapping support sets—since small variations in the underlying gating scores $s_{j,i}$ can theoretically yield disjoint subsets despite smooth biases—it empirically drives high support overlap between adjacent phases (i.e., $J(\mathcal{C}_\varphi, \mathcal{C}_{\varphi\pm1}) \to$ high, where J denotes the Jaccard similarity). Therefore, as the scheduler minimizes $\Delta\varphi$, the batch union practically converges toward K_{hard}. While not an absolute mathematical lock, this empirical convergence confirms that HBM traffic remains restricted to a localized footprint, effectively circumventing the continuous batching Expert Explosion.

C Experimental Setup, Hardware, and A/B Testing Methodology

This section details the infrastructure, dataset configurations, and hyperparameters utilized to evaluate the Phase-Constrained MoE architecture. To ensure a rigorous evaluation of our proposed routing constraints, we implemented a strict A/B testing methodology designed to isolate structural routing effects from data-driven confounding variables.

C.1 A/B Testing Justification and the Performance Gap

As reported in Section 5.2, our baseline and Phase-MoE variants exhibit a measurable generative performance gap when compared to state-of-the-art frontier discrete diffusion models such as LLaDA 2.0 (Bie et al., 2025). This gap is fully anticipated and stems from deliberate simplifications made to our training pipeline to isolate the structural impact of the MoE routing mechanisms:

1. **Token Volume and Data Diversity:** While frontier models like `LLaDa2.0-mini` (Bie et al., 2025) also initialize from `Ling-mini-2.0` (Team et al., 2025), they undergo Continual Pre-Training (CPT) on curated, diverse datasets exceeding 100B tokens. To maintain a strictly controlled experimental environment, we constrained our total training budget to approximately 12B tokens per model variant: exactly 10B tokens on a single CPT corpus (SmolLM-Corpus (Ben Allal et al., 2024)) and 2B tokens on a single Supervised Fine-Tuning (SFT) corpus (OpenHermes2.5 (Teknium, 2023)).
2. **Omission of Post-Training Enhancements:** We explicitly disabled advanced post-training interventions utilized by frontier models, including Direct Preference Optimization (DPO), Reinforcement Learning (RL) alignment, and the Auxiliary Confidence Loss designed to artificially sharpen parallel decoding predictions.

By stripping away these complex, data-heavy pipelines, we mathematically guarantee that any observed variations in High-Bandwidth Memory (HBM) traffic, expert utilization, and generation accuracy are strictly attributable to the presence or absence of the Phase-Constrained MoE architecture.

Furthermore, this strictly controlled environment contextualizes the internal performance delta between our unconstrained Baseline and the aggressively pruned Phase-MoE (Top-96). While Top-96 experiences notable regressions in multi-step reasoning and formatting (e.g., GSM8K, IFEval), this isolates the raw capacity tax required to enforce a hard $K_{\text{hard}} = 96$ memory bound. Without the compensating effects of massive scale ($>$ 100B tokens) or reinforcement learning to smooth over routing constraints, these metrics honestly reflect the baseline architectural trade-off between absolute generative fidelity and production-scale memory efficiency.

C.2 Hardware Infrastructure

All training phases (CPT and SFT) for the baseline and Phase-MoE models were executed on a single compute node equipped with 8 NVIDIA H100 GPUs. The training environment leveraged the `dFactory` framework (InclusionAI, 2025), utilizing Fully Sharded Data Parallelism (FSDP2) combined with Expert Parallelism (EP=4) to distribute the MoE state effectively. All downstream generative evaluations, inference latency profiling, and continuous batching simulations were conducted on a single NVIDIA A100 (80GB) GPU using the `dInfer` (Ma et al., 2025) serving engine.

Table 3: **Comprehensive Training and Routing Hyperparameters.** Configurations used across the baseline and Phase-MoE variants for Continual Pre-Training (CPT) and Supervised Fine-Tuning (SFT).

Category	Parameter	Value
Optimization	Optimizer	AdamW
	Peak Learning Rate	1.0×10^{-5}
	LR Schedule	Cosine with 5% Warmup
	Weight Decay	0.1
	Max Gradient Norm	1.0
	Global Batch Size	64
Diffusion Dynamics	Target Block Size (B)	32
	Sequence Length	2048
Phase-MoE Routing	Load Balancing Push (λ_{lb})	0.001
	Phase Specialization Pull (λ_{spec})	0.001
	Gaussian Smoothing σ (for $B = 32$)	1.5
	Gaussian Kernel Window w (for $B = 32$)	5

C.3 Hyperparameter Configurations and Gaussian Scaling

Table 3 outlines the core hyperparameters utilized during the CPT and SFT phases.

Scaling the Gaussian Smoothing Kernel: During the Soft Shaping stage, the 1D Gaussian blur applied to the phase dimension ensures that adjacent temporal phases share overlapping expert support. The exact shape of this kernel is robust and not highly sensitive to hyperparameter tuning, provided it resembles a standard normal distribution. However, its spatial parameters must scale relative to the parallel block size B (which dictates the total number of discrete phases).

For our primary setting of $B = 32$, we utilize $\sigma = 1.5$ and a kernel window of $w = 5$. When scaling down the block size to $B = 16$, we proportionally compress the smoothing radius ($\sigma = 1.0, w = 3$) to prevent excessive expert smearing across the shortened denoising trajectory. Conversely, for fine-grained phase resolutions ($B = 64$), we scale the kernel up ($\sigma = 3.0, w = 9$) to guarantee sufficient support overlap across the highly discretized temporal bins.

D Additional Empirical Results for Section 5

This appendix provides extended results and structural diagnostics complementing the experimental analysis in Section 5. We present single-GPU hardware efficiency benchmarks under isolated single-sequence execution, analyze global expert lifespan distributions to verify routing health, and track the temporal evolution of phase specialization during router training.

D.1 Single-Sequence Hardware Efficiency

Table 4: **Single GPU Hardware Efficiency** ($B_{\text{step}} = 1$, parallel block size $B = 32$). Measurements are averaged across generative benchmarks (GSM8K, HumanEval, IFEval). APF (Activated experts Per Forward pass) measures the number of unique experts fetched from HBM out of the total $N = 256$ pool.

Model Configuration	APF (Prefill)	APF (Decode)	MoE Latency (ms)	Latency Reduction
Baseline ($B = 32$)	77.47	60.13	9.27	–
Phase-MoE (Top-128)	80.26	45.67	8.50	8.3%
Phase-MoE (Top-96)	79.35	40.93	8.08	12.8%

As reported in Table 4, the baseline model exhibits severe expert dispersion: parallelizing 32 tokens causes the union to fetch over 60 unique experts per step. By enforcing hard support pools, Phase-MoE (Top-96) curtails Decode APF to 40.93, yielding a clean 12.8% speedup in the memory-bound MoE kernel. Crucially, the Prefill APF remains consistently high ($\sim$77–80) across all models.

This confirms that isolating Phase 0 correctly preserves maximum representational capacity during compute-bound context encoding while strictly compressing HBM traffic during memory-bound active denoising.

D.2 Temporal Specialization Dynamics

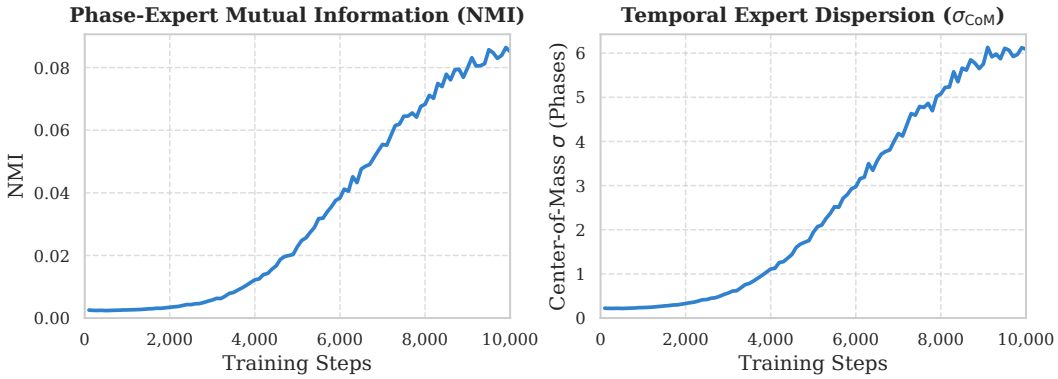


Figure 6: **Evolution of phase–expert specialization across training steps (representative layer). Left:** Normalized Mutual Information (NMI) between active expert activations and mask density phases ($\varphi \in \{1, \ldots, B\}$). **Right:** Token-weighted standard deviation of expert temporal centers of mass (σ_{CoM}).

To empirically validate the Soft Shaping curriculum introduced in Section 3.2, we track the temporal evolution of expert routing dynamics over the first 10,000 soft training steps.

As illustrated in Figure 6 (left), the Normalized Mutual Information (NMI) between active expert allocations and discrete mask density phases starts near zero upon initialization. As training progresses, the NMI steadily increases, confirming that the router organically learns strong, data-driven phase affinities rather than collapsing into a static, phase-agnostic distribution.

Simultaneously, Figure 6 (right) tracks temporal expert dispersion via the token-weighted standard deviation of the experts' temporal centers of mass (σ_{CoM}). A persistently low σ_{CoM} would indicate catastrophic temporal clustering (e.g., all experts clustering exclusively in early global planning). Instead, we observe a smooth spatial expansion that stabilizes as experts smoothly specialize across distinct denoising phases of iterative decoding. This verifies that the Soft Shaping phase gently coerces the expert pool into a well-distributed, temporally specialized organization, safely preparing the architecture for Hard Pruning.

E Comprehensive Evaluation of Prior Baselines under Continuous Batching

Expanding upon the empirical summary provided in Section 5.3, this section details the continuous batching evaluation methodology, the integration mechanics for post-hoc routing interventions, and an exhaustive quantitative analysis of the *Continuous Batching Dilemma*. To explicitly isolate and evaluate the structural limitations of inference-time mitigations under production workloads, we evaluate three recent state-of-the-art baselines: Dynamic Expert Sharing (DES) (Chen et al., 2026), TEAM (Wei et al., 2026), and dMoE (Feng et al., 2026).

E.1 Experimental Setup: Mixed Workloads and "Ghost Sequences"

Standard generative benchmarks typically evaluate isolated, single-sequence generation. However, simulating a production continuous batching environment requires capturing both semantic heterogeneity (diverse task domains) and temporal heterogeneity (varying denoising phases) under a sustained hardware load.

Table 5: **Comprehensive Generative Performance under Continuous Batching Heterogeneity.** Downstream evaluation of the unconstrained baseline, prior routing interventions (DES, TEAM, dMoE), and Phase-MoE across varying step batch sizes ($B_{\text{step}} \in \{1, 8, 16\}$) at concurrency $C = 64$. These results empirically validate the *Continuous Batching Dilemma*. While post-hoc methods preserve accuracy in isolated single-sequence settings ($B_{\text{step}} = 1$), they face a strict Pareto trade-off under batching. Preserving sequence-level routing independence (e.g., DES-SEQ, dMoE-SEQ) maintains baseline accuracy but fails to constrain memory traffic (see Table 6). Conversely, forcing a batch-level consensus (e.g., DES-BATCH, dMoE-BATCH) to cap memory fetches triggers significant semantic collapse, degrading reasoning (GSM8K) and formatting (IFEval) accuracy. Phase-MoE breaks this trade-off, preserving competitive accuracy while structurally bounding hardware costs.

Configuration	ARC-C	MMLU	HellaSwag	PIQA	GSM8K	IFEval
BASELINE						
FCFS ($C = 64, B_{\text{step}} = 1$)	83.00%	62.50%	68.00%	79.00%	64.00%	49.00%
FCFS ($C = 64, B_{\text{step}} = 8$)	84.00%	63.50%	68.00%	79.00%	62.00%	51.67%
PA ($C = 64, B_{\text{step}} = 8$)	84.50%	62.50%	67.00%	78.50%	66.00%	51.67%
FCFS ($C = 64, B_{\text{step}} = 16$)	84.00%	62.00%	66.50%	80.50%	64.00%	54.33%
PA ($C = 64, B_{\text{step}} = 16$)	83.50%	62.50%	67.50%	78.50%	68.00%	52.00%
BASELINE + DES						
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 1$)	84.50%	61.00%	67.50%	79.50%	62.00%	50.33%
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 8$)	83.50%	62.50%	68.50%	80.50%	60.00%	49.33%
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 16$)	83.50%	61.50%	68.00%	80.00%	64.00%	49.67%
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 1$)	83.50%	63.00%	68.50%	80.00%	66.00%	52.67%
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 8$)	83.50%	61.00%	68.50%	79.00%	66.00%	49.67%
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 16$)	82.50%	62.00%	67.50%	80.00%	66.00%	53.67%
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 1$)	83.00%	62.00%	68.50%	80.00%	68.00%	51.00%
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 8$)	82.50%	62.00%	67.00%	80.50%	66.00%	49.33%
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 16$)	83.00%	62.50%	67.50%	79.00%	68.00%	52.33%
DES-BATCH ($M = 128, C = 64, B_{\text{step}} = 8$)	84.00%	61.50%	67.50%	80.50%	64.00%	48.33%
DES-BATCH ($M = 128, C = 64, B_{\text{step}} = 16$)	83.50%	59.00%	67.00%	81.00%	64.00%	47.00%
DES-BATCH ($M = 96, C = 64, B_{\text{step}} = 8$)	82.50%	60.00%	67.00%	80.50%	64.00%	46.67%
DES-BATCH ($M = 96, C = 64, B_{\text{step}} = 16$)	82.50%	59.50%	68.50%	78.00%	60.00%	44.00%
DES-BATCH ($M = 64, C = 64, B_{\text{step}} = 8$)	81.50%	59.00%	68.00%	79.00%	62.00%	43.00%
DES-BATCH ($M = 64, C = 64, B_{\text{step}} = 16$)	82.00%	60.00%	66.00%	80.00%	60.00%	42.33%
BASELINE + TEAM						
LAC-SEQ ($C = 64, B_{\text{step}} = 1$)	83.50%	62.00%	66.50%	78.00%	64.00%	48.00%
LAC-SEQ ($C = 64, B_{\text{step}} = 8$)	83.50%	62.50%	66.00%	80.00%	66.00%	48.00%
LAC-SEQ ($C = 64, B_{\text{step}} = 16$)	83.00%	61.50%	65.50%	79.50%	64.00%	47.00%
LAC-BATCH ($C = 64, B_{\text{step}} = 8$)	84.00%	62.00%	65.50%	80.00%	68.00%	48.33%
LAC-BATCH ($C = 64, B_{\text{step}} = 16$)	83.50%	61.00%	65.00%	79.50%	64.00%	49.33%
BASELINE + dMoE						
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 1$)	84.50%	62.50%	67.50%	80.00%	64.00%	49.33%
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 8$)	83.50%	60.50%	68.00%	80.50%	66.00%	48.67%
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 16$)	83.50%	60.50%	66.50%	81.50%	64.00%	49.33%
dMoE-BATCH ($P = 0.6, C = 64, B_{\text{step}} = 8$)	82.50%	57.50%	68.00%	78.50%	52.00%	43.00%
dMoE-BATCH ($P = 0.6, C = 64, B_{\text{step}} = 16$)	82.50%	60.00%	67.00%	79.50%	56.00%	40.00%
PHASE-MOE (TOP-128)						
FCFS ($C = 64, B_{\text{step}} = 1$)	82.00%	58.50%	68.00%	78.00%	58.00%	52.00%
FCFS ($C = 64, B_{\text{step}} = 8$)	82.00%	58.50%	67.50%	79.00%	68.00%	49.00%
PA ($C = 64, B_{\text{step}} = 8$)	81.00%	59.50%	68.00%	79.00%	62.00%	50.00%
FCFS ($C = 64, B_{\text{step}} = 16$)	83.00%	58.00%	68.00%	78.00%	62.00%	49.00%
PA ($C = 64, B_{\text{step}} = 16$)	81.50%	59.00%	68.00%	78.50%	66.00%	51.00%
PHASE-MOE (TOP-96)						
FCFS ($C = 64, B_{\text{step}} = 1$)	79.50%	58.00%	65.50%	77.50%	62.00%	47.00%
FCFS ($C = 64, B_{\text{step}} = 8$)	80.00%	58.00%	65.50%	76.50%	64.00%	47.67%
PA ($C = 64, B_{\text{step}} = 8$)	79.00%	56.50%	66.50%	77.50%	56.00%	50.33%
FCFS ($C = 64, B_{\text{step}} = 16$)	81.50%	57.50%	65.50%	77.50%	64.00%	48.00%
PA ($C = 64, B_{\text{step}} = 16$)	81.00%	56.50%	64.50%	77.00%	62.00%	54.00%

Table 6: **Hardware Efficiency, Memory Traffic, and Routing Overhead.** Evaluation of unique expert activations (U), MoE kernel latency, temporal phase spread ($\Delta\varphi$), and CPU/GPU routing overhead. In single-sequence isolation ($B_{\text{step}} = 1$), inference-time heuristics successfully cull unique experts, occasionally surpassing Phase-MoE's structural bounds (e.g., TEAM achieves $U = 27.81$). However, under production continuous batching ($B_{\text{step}} > 1$), sequence-level baselines (DES-SEQ, LAC-SEQ, dMoE-SEQ) suffer an *Expert Explosion*, rapidly saturating HBM bandwidth ($U > 150$). *Note on Overhead:* Baseline routing overheads (3.7–9.0 ms) are profiled using standard PyTorch implementations. While custom hardware-specific fused CUDA kernels could theoretically mask this latency tax, eliminating overhead does not resolve their fundamental mathematical inability to simultaneously bound batch-level memory traffic and preserve semantic accuracy. Phase-MoE achieves optimal throughput with negligible scheduling overhead (≤ 0.03 ms).

Configuration	Unique Experts (U)	MoE Lat (ms)	Phase Spread	Overhead (ms)
BASELINE				
FCFS ($C = 64, B_{\text{step}} = 1$)	73.90	9.86	0.00	–
FCFS ($C = 64, B_{\text{step}} = 8$)	203.91	20.21	25.49	–
PA ($C = 64, B_{\text{step}} = 8$)	190.53	19.18	2.26	0.023
FCFS ($C = 64, B_{\text{step}} = 16$)	228.76	28.21	28.65	–
PA ($C = 64, B_{\text{step}} = 16$)	216.67	27.12	5.77	0.022
BASELINE + DES				
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 1$)	64.27	9.28	0.00	3.852
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 8$)	184.52	18.71	25.55	4.137
DES-SEQ ($M = 128, C = 64, B_{\text{step}} = 16$)	211.65	26.65	28.60	4.115
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 1$)	60.01	9.14	0.00	3.964
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 8$)	177.97	18.26	25.44	4.154
DES-SEQ ($M = 96, C = 64, B_{\text{step}} = 16$)	206.50	26.24	28.61	4.227
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 1$)	50.89	8.34	0.00	3.812
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 8$)	162.43	17.02	25.50	4.042
DES-SEQ ($M = 64, C = 64, B_{\text{step}} = 16$)	193.86	25.04	28.74	4.138
DES-BATCH ($M = 128, C = 64, B_{\text{step}} = 8$)	125.18	14.46	25.44	3.782
DES-BATCH ($M = 128, C = 64, B_{\text{step}} = 16$)	127.15	18.92	28.53	3.717
DES-BATCH ($M = 96, C = 64, B_{\text{step}} = 8$)	94.90	12.55	25.25	3.743
DES-BATCH ($M = 96, C = 64, B_{\text{step}} = 16$)	95.66	16.01	28.61	3.912
DES-BATCH ($M = 64, C = 64, B_{\text{step}} = 8$)	63.62	10.88	25.46	3.704
DES-BATCH ($M = 64, C = 64, B_{\text{step}} = 16$)	63.85	12.81	28.59	3.798
BASELINE + TEAM				
LAC-SEQ ($C = 64, B_{\text{step}} = 1$)	27.81	9.56	0.00	4.277
LAC-SEQ ($C = 64, B_{\text{step}} = 8$)	112.07	19.86	25.09	4.280
LAC-SEQ ($C = 64, B_{\text{step}} = 16$)	150.08	28.10	28.70	4.492
LAC-BATCH ($C = 64, B_{\text{step}} = 8$)	110.40	19.89	25.76	4.165
LAC-BATCH ($C = 64, B_{\text{step}} = 16$)	148.26	28.02	28.66	4.176
BASELINE + dMoE				
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 1$)	18.74	7.40	0.00	8.400
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 8$)	90.48	12.29	25.59	9.077
dMoE-SEQ ($P = 0.6, C = 64, B_{\text{step}} = 16$)	129.12	19.12	28.74	9.010
dMoE-BATCH ($P = 0.6, C = 64, B_{\text{step}} = 8$)	39.74	10.05	25.49	7.750
dMoE-BATCH ($P = 0.6, C = 64, B_{\text{step}} = 16$)	46.82	11.24	28.62	7.869
PHASE-MOE (TOP-128)				
FCFS ($C = 64, B_{\text{step}} = 1$)	54.88	8.73	0.00	–
FCFS ($C = 64, B_{\text{step}} = 8$)	194.79	19.46	25.08	–
PA ($C = 64, B_{\text{step}} = 8$)	118.37	14.18	2.17	0.024
FCFS ($C = 64, B_{\text{step}} = 16$)	229.36	28.30	28.47	–
PA ($C = 64, B_{\text{step}} = 16$)	141.70	20.35	5.66	0.023
PHASE-MOE (TOP-96)				
FCFS ($C = 64, B_{\text{step}} = 1$)	49.86	8.34	0.00	–
FCFS ($C = 64, B_{\text{step}} = 8$)	177.66	18.32	25.50	–
PA ($C = 64, B_{\text{step}} = 8$)	95.39	12.66	2.22	0.024
FCFS ($C = 64, B_{\text{step}} = 16$)	210.20	26.66	28.57	–
PA ($C = 64, B_{\text{step}} = 16$)	104.52	17.79	5.59	0.023

Mixed Workload: To reflect real-world serving diversity, our evaluation dataset is constructed by mixing prompts across multiple distinct domains: mathematical reasoning (GSM8K (Cobbe et al., 2021)), knowledge retrieval (MMLU (Hendrycks et al., 2020), ARC-C (Clark et al., 2018)), commonsense reasoning (HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020)), and instruction following (IFEval (Zhou et al., 2023)). This guarantees that concurrent requests within an execution batch naturally demand semantically disjoint expert subsets.

The "Ghost Sequence" Concurrency Mechanism: To rigorously evaluate hardware efficiency alongside generative accuracy, the execution engine must maintain a strictly constant concurrency level (C). If sequences are not immediately replaced upon completion, the effective batch size artificially dwindles toward the end of the workload, corrupting the steady-state hardware telemetry (latency and HBM traffic). To resolve this, we employ a *Ghost Sequence* replenishment strategy.

We initialize the execution pool with C real evaluation prompts. Whenever a sequence emits an EOS token and completes, its output is logged for downstream generative evaluation. To maintain exact concurrency, the scheduler immediately injects the next real request from the workload pool. Once the finite pool of real requests is fully exhausted, the scheduler begins injecting *ghost sequences*—randomly sampled prompts from the dataset—to replace finished sequences. The engine continues computing forward passes for these ghost sequences to maintain constant hardware pressure and intra-batch heterogeneity, but their textual outputs are discarded. This guarantees that all reported generative accuracies for real requests are measured under the exact, sustained memory bandwidth pressure of C concurrent requests.

E.2 Integration of Prior Routing Interventions

We adapt three recent MoE routing baselines to our continuous batching engine. Because continuous batching processes B_{step} asynchronous sequences simultaneously, any block-level or sequence-level heuristic faces an implementation fork: apply the heuristic independently per sequence (*-SEQ*), or force a global consensus across the execution step batch (*-BATCH*). We evaluate both paradigms for each baseline:

- **Dynamic Expert Sharing (DES):** A training-free method that identifies a compact coreset of experts. We implement their flagship Saliency-Aware Voting algorithm (DES-Vote), which controls the coreset size via a budget parameter M. To evaluate this mechanism under continuous batching, we apply it in two ways: **DES-SEQ**, which calculates the vote-based coreset independently for each sequence, and **DES-BATCH**, which forces a unified voting consensus across all B_{step} sequences to cap global memory traffic. We sweep the budget constraint across $M \in \{64, 96, 128\}$.
- **TEAM:** We isolate their Limited Activation for Cold Tokens (LAC) mechanism—evaluating both **LAC-SEQ** (local) and **LAC-BATCH** (global) variants—which restricts *cold* tokens to experts activated by *hot* tokens. *We explicitly disable Speculative Exploration (SEH). While 4-branch speculation absorbs idle compute at $B_{\text{step}} = 1$, deploying it across $B_{\text{step}} \geq 16$ concurrent sequences exponentially inflates active token volume. This triggers compute bottlenecks under continuous batching.*
- **dMoE:** Unlike DES and TEAM (which are training-free inference interventions), dMoE requires a distinct fine-tuned checkpoint. We fine-tune the base model using their block-level self-distillation pipeline (top-p thresholding) to create a dedicated dMoE baseline. During continuous batching inference, we evaluate **dMoE-SEQ**, which applies top-p filtering independently per sequence (operating in-distribution to its single-sequence training setup), and **dMoE-BATCH**, which applies a global top-p filter across the entire asynchronous batch (operating out-of-distribution to its training, but necessary to force a cap on total HBM traffic).

E.3 Performance in Isolated Execution ($B_{\text{step}} = 1$)

When evaluated in pure isolation (execution step batch size $B_{\text{step}} = 1$, where the engine sequentially dispatches a single request from the active pool of $C = 64$), post-hoc interventions demonstrate remarkable efficacy. Because a single sequence is semantically uniform and temporally homogenous (all tokens reside at the exact same denoising phase φ), runtime heuristics successfully cull redundant experts with minimal accuracy loss.

As visually anchored in Figure 5a, and detailed in Tables 5 and 6, these baseline methods frequently match or even surpass Phase-MoE's structural bounds in single-sequence isolation. For instance, at $B_{\text{step}} = 1$, TEAM (LAC-SEQ) compresses the unique expert count to just $U = 27.81$, and dMoE-SEQ achieves $U = 18.74$. In contrast, Phase-MoE (Top-96) fetches 49.86 experts. Because Phase-MoE is architecturally trained to provide a robust, fixed-size candidate pool ($\mathcal{C}_\varphi$) ensuring steady generative quality, it does not aggressively cull experts on-the-fly, resulting in a higher fetch rate in strict single-batch scenarios.

However, this single-sequence efficacy does not seamlessly translate to the multi-sequence environments required for production serving. The advantages of these post-hoc methods degrade significantly when execution scales to handle multi-sequence heterogeneity.

E.4 The Continuous Batching Dilemma ($B_{\text{step}} > 1$)

Under continuous batching, asynchronous sequences arrive at arbitrary denoising phases ($\Delta\varphi \gg 0$) and span disparate semantic domains. Within this environment, post-hoc interventions face a conflict:

The Independence Trap (-SEQ Variants): If the intervention preserves sequence-level independence to maintain generative accuracy (DES-SEQ, LAC-SEQ, dMoE-SEQ), it fundamentally fails to bound batch-level memory traffic. Because concurrent requests are semantically and temporally disjoint, their independently calculated coresets or top-p subsets have near-zero overlap. Consequently, the union of these subsets rapidly re-saturates the HBM bandwidth as B_{step} scales.

- At $C = 64$, $B_{\text{step}} = 16$, DES-SEQ ($M = 96$) escalates to $U = 206.50$ unique experts—offering virtually no relief compared to the unconstrained baseline's $U = 228.76$.
- Similarly, dMoE-SEQ expands from $U = 18.74$ (at $B_{\text{step}} = 1$) back to a prohibitive $U = 129.12$ (at $B_{\text{step}} = 16$).

The Consensus Trap (-BATCH Variants): To circumvent this memory expansion, one can force the heuristics to calculate a single, global consensus across the entire continuous batch (DES-BATCH, dMoE-BATCH). While this successfully caps HBM traffic (e.g., DES-BATCH $M = 96$ stabilizes at $U = 95.66$), it triggers severe degradation in generative reasoning and instruction following. Forcing temporally disparate and semantically unrelated sequences into an artificial routing bottleneck deprives the model of its required representational capacity.

- DES-BATCH ($M = 96$) causes IFEval formatting accuracy to drop from the baseline's 54.33% down to 44.00%.
- dMoE-BATCH suffers similar degradation, dropping GSM8K accuracy from 64.00% to 56.00% and IFEval down to 40.00%. Because the official dMoE checkpoint was explicitly distilled on a heavily skewed mixture of 700K synthetic reasoning and mathematical traces, its routing distribution is artificially overfitted to the reasoning domain. When subjected to standard diverse distillation and evaluation process under the forced consensus of a semantically heterogeneous batch, its aggregated routing results in semantic collapse.

Breaking the Dilemma: Phase-MoE resolves this bottleneck structurally. Because the Phase-MoE router is trained via a soft-to-hard curriculum to naturally overlap expert support sets for adjacent denoising phases, our Phase-Aware CPU scheduler can tightly cluster asynchronous sequences ($\Delta\varphi \to 0$) *without* forcing an artificial inference-time consensus. This synergy preserves high accuracy (IFEval 54.00%) while structurally bounding memory traffic ($U = 104.52$ for Top-96 at $B_{\text{step}} = 16$).

The "Exposure Time" Penalty for Long-Form Generation. A closer inspection of Table 5 reveals a stark divergence in how batch-consensus mitigations (DES-BATCH, dMoE-BATCH) degrade performance across different task topologies. Multiple-choice (MC) benchmarks (ARC-C, MMLU) experience relatively mild degradation, whereas long-form generative tasks (GSM8K, IFEval) suffer catastrophic collapse. For example, at $B_{\text{step}} = 16$, dMoE-BATCH drops MMLU by only 2.0% against the baseline, but plummets on GSM8K (-8.0%) and IFEval (-14.33%).

This discrepancy is driven by the sequence's *exposure time* to batch-level heterogeneity. MC tasks require generating only a handful of tokens; they clear the execution queue rapidly, minimizing their exposure to forced routing compromises. In contrast, generative tasks require significantly more

iterative decoding steps. Under continuous batching, these long-lived sequences share the execution batch with a constantly rotating cast of asynchronous requests spanning disparate domains. Under forced consensus (-BATCH), a long-form GSM8K sequence must compromise its ideal expert routing to accommodate the semantic needs of the rest of the batch at each step.

E.5 The Mathematical Tautology of TEAM's Hot-Cold Routing Constraint

As observed in Table 6, both sequence-level and batch-level applications of TEAM's Limited Activation for Cold Tokens (LAC) fail to bound memory traffic under continuous batching. We can formally prove that this failure is structural, as LAC-SEQ and LAC-BATCH are mathematically equivalent in their upper bound for HBM traffic.

Let $\mathcal{B} = \{1, 2, \ldots, B_{\text{step}}\}$ denote the execution batch. TEAM partitions the active tokens of sequence m into a Hot set H_m, a Newly Decoded set D_m, and a Cold set C_m. Let $R(t)$ denote the unconstrained Top-k routing selection for token t.

Under **LAC-SEQ**, the necessary expert pool for sequence m is defined by its hot and newly decoded tokens: $E_{\text{nec}}^{(m)} = \bigcup_{t \in H_m \cup D_m} R(t)$. Because cold tokens C_m are restricted to routing exclusively within $E_{\text{nec}}^{(m)}$, they contribute zero additional unique experts to the sequence. The total batch-level HBM traffic is the union of these sequence-level pools:

$$U_{\text{SEQ}} = \left| \bigcup_{m \in \mathcal{B}} E_{\text{nec}}^{(m)} \right| = \left| \bigcup_{m \in \mathcal{B}} \bigcup_{t \in H_m \cup D_m} R(t) \right| \tag{22}$$

Under **LAC-BATCH**, the necessary expert pool is aggregated globally across all sequences in the step batch: $E_{\text{nec}}^{(\text{batch})} = \bigcup_{m \in \mathcal{B}} \bigcup_{t \in H_m \cup D_m} R(t)$. Cold tokens for all sequences are then routed within this shared global pool, again fetching zero additional experts outside of it. The total batch-level HBM traffic is exactly the cardinality of this global set:

$$U_{\text{BATCH}} = \left| E_{\text{nec}}^{(\text{batch})} \right| = \left| \bigcup_{m \in \mathcal{B}} \bigcup_{t \in H_m \cup D_m} R(t) \right| \tag{23}$$

Consequently, $U_{\text{SEQ}} \equiv U_{\text{BATCH}}$. The memory footprint is identical, demonstrating that TEAM's heuristic cannot scale to multi-sequence continuous batching without inducing an Expert Explosion.

E.6 Computational Overhead and the Custom Kernel Context

As reported in Table 6, post-hoc inference interventions incur substantial computational overhead in standard PyTorch implementations. Operations such as calculating spatial distances and executing double-round routing (TEAM), or performing token score aggregation and dynamic top-p sorting (dMoE, DES), introduce CPU/GPU synchronization penalties ranging from 3.7 ms to 9.0 ms per step.

It must be acknowledged that in highly optimized production environments, hardware-specific fused CUDA kernels could theoretically be deployed to mask this computational tax, bringing the runtime overhead closer to zero. *However, eliminating this execution overhead does not resolve the Continuous Batching Dilemma.* Even assuming a 0.0 ms computational penalty, inference-time heuristics remain trapped between the memory explosion of independent routing and the representational collapse of forced batch consensus.

F Comprehensive Continuous Batching Results

This section provides the complete empirical results from our continuous batching simulation engine, expanding upon the summarized findings in Section 5.4. We evaluate the unconstrained Baseline, Phase-MoE (Top-128), and Phase-MoE (Top-96) across an exhaustive grid of active request concurrencies ($C \in \{16, 32, 64, 128\}$) and execution step batch sizes ($B_{\text{step}} \in \{4, 8, 16, 32\}$).

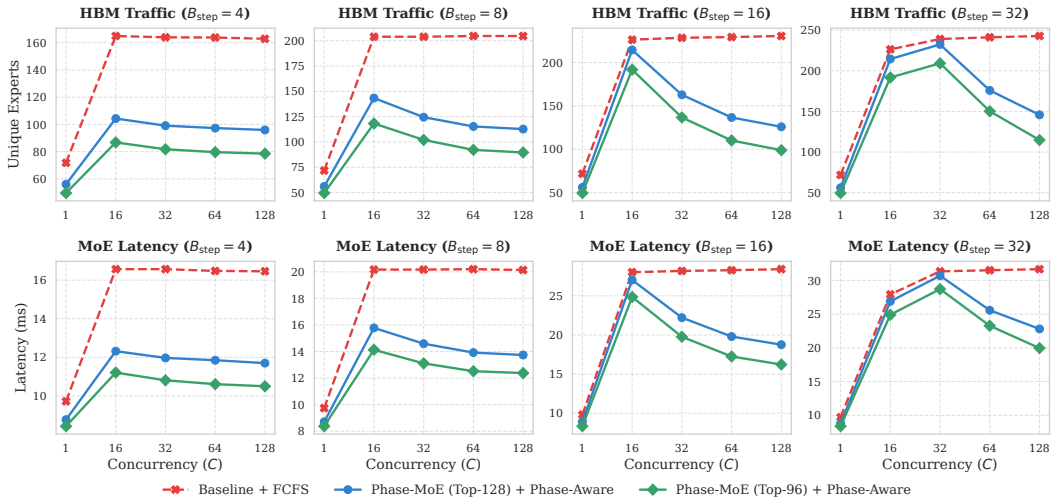


Figure 7: **Comprehensive continuous batching scalability across all execution step batch sizes ($B_{\text{step}} \in \{4, 8, 16, 32\}$) and request concurrencies ($C \in \{1, 16, 32, 64, 128\}$). Top row:** Number of unique expert weights fetched from High-Bandwidth Memory (HBM) per forward pass during iterative block decoding. **Bottom row:** Corresponding mean MoE kernel execution latency in milliseconds. Across all step batch sizes, the unconstrained baseline quickly saturates global memory bandwidth as concurrency scales, whereas Phase-MoE with Phase-Aware scheduling consistently bounds support sparsity and maintains flat latency profiles as concurrency increases.

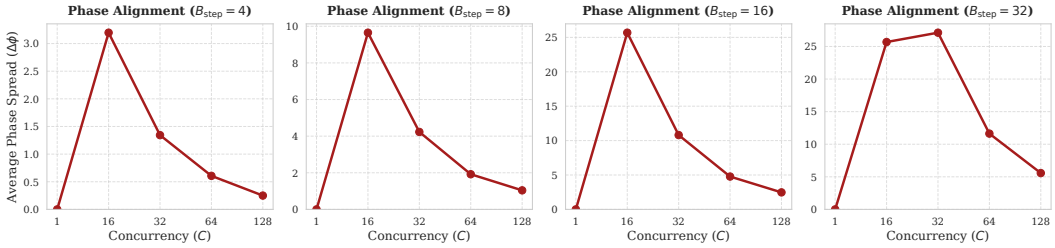


Figure 8: **Phase-Aware Scheduler Alignment Dynamics across Batch Scales.** Mean phase spread ($\Delta\varphi = \max(\varphi) - \min(\varphi)$) within concurrently executed decode batches across request concurrencies ($C \in \{1, 16, 32, 64, 128\}$) and execution step batch sizes ($B_{\text{step}} \in \{4, 8, 16, 32\}$). At $C = 1$, $\Delta\varphi$ is trivially zero. At low concurrency ($C = 16$), the scheduler is forced to batch across wide temporal gaps to maintain throughput. As concurrency scales ($C \to 128$), the richer request candidate pool enables the sliding-window scheduler to achieve tight phase clustering ($\Delta\varphi \to 0$), directly translating training-time Phase-Constrained MoE bounds into near-optimal HBM bandwidth savings during inference.

Table 7: **Comprehensive Continuous Batching Simulation** (Total Expert Pool $N = 256$). Detailed evaluation of the unconstrained Baseline, Phase-MoE (Top-128), and Phase-MoE (Top-96) across all execution batch step sizes ($B_{\text{step}} \in \{4, 8, 16, 32\}$) and request concurrencies ($C \in \{16, 32, 64, 128\}$) under First-Come-First-Serve (FCFS) and Phase-Aware (PA) scheduling. Here, $\Delta\varphi$ denotes the mean phase spread (the maximum gap in block mask density among concurrently batched sequences), *Experts* (U) represents the mean number of unique expert weights transferred from HBM to SRAM per forward step, and *Lat.* (Latency) is the mean MoE kernel execution time in milliseconds (ms). **Bold** and underline denote the best and second-best results for memory traffic and latency within each (C, B_{step}) setting, respectively. The configuration $B_{\text{step}} = 32$ at $C = 16$ is omitted (–) as concurrency is strictly lower than the execution step batch size.

			$B_{\text{step}} = 4$			$B_{\text{step}} = 8$			$B_{\text{step}} = 16$			$B_{\text{step}} = 32$		
C	**Model**	**Sched.**	$\Delta\varphi$	**Experts**	**Lat.**	$\Delta\varphi$	**Experts**	**Lat.**	$\Delta\varphi$	**Experts**	**Lat.**	$\Delta\varphi$	**Experts**	**Lat.**
16	Baseline	FCFS	19.28	164.95	16.57	24.37	203.98	20.17	26.61	226.40	28.01	–	–	–
		PA	3.27	153.82	15.80	10.02	194.59	19.44	26.61	226.40	27.97	–	–	–
	Top-128	FCFS	18.05	139.99	14.77	23.50	187.86	19.01	25.45	214.53	26.90	–	–	–
		PA	3.23	104.31	12.32	9.90	143.37	15.78	25.45	214.53	27.02	–	–	–
	Top-96	FCFS	18.71	127.25	13.85	23.70	168.81	17.55	24.99	191.80	24.85	–	–	–
		PA	3.09	**86.76**	**11.21**	9.03	**118.24**	**14.13**	24.99	**191.80**	**24.85**	–	–	–
32	Baseline	FCFS	19.42	163.99	16.57	24.53	203.92	20.17	27.47	228.56	28.18	27.93	239.15	31.34
		PA	1.36	151.87	15.68	4.38	190.81	19.25	11.25	219.76	27.42	27.93	239.15	31.33
	Top-128	FCFS	18.73	141.59	14.93	24.37	191.02	19.19	26.87	221.31	27.61	27.03	232.45	30.76
		PA	1.33	99.06	11.97	4.08	124.48	14.59	10.93	162.81	22.21	27.03	232.45	30.69
	Top-96	FCFS	18.84	126.68	13.96	24.40	172.01	17.85	26.82	200.31	25.68	26.39	209.21	**28.61**
		PA	1.34	**81.72**	**10.81**	4.23	**102.05**	**13.11**	10.23	**136.69**	**19.77**	26.39	**209.21**	28.71
64	Baseline	FCFS	19.12	163.84	16.48	24.91	204.67	20.20	28.01	229.38	28.27	29.11	241.25	31.50
		PA	0.63	151.13	15.63	2.00	187.76	19.02	5.02	215.92	26.99	11.97	235.25	31.07
	Top-128	FCFS	19.11	142.94	15.08	24.85	192.90	19.33	27.64	225.21	27.93	28.56	239.46	31.35
		PA	0.60	97.29	11.85	1.89	115.34	13.92	4.60	136.66	19.80	11.51	175.85	25.57
	Top-96	FCFS	18.76	126.87	13.87	24.76	173.02	17.79	27.66	204.39	26.07	28.18	218.33	29.50
		PA	0.59	**79.62**	**10.61**	1.87	**92.20**	**12.52**	4.70	**110.19**	**17.25**	11.41	**150.24**	**23.27**
128	Baseline	FCFS	19.51	162.92	16.46	24.97	204.67	20.14	28.35	230.60	28.40	29.68	242.84	31.66
		PA	0.26	149.97	15.59	1.05	187.65	19.07	2.62	214.87	26.96	6.03	232.58	30.80
	Top-128	FCFS	19.41	142.86	14.99	24.89	192.37	19.28	28.03	226.94	28.06	29.34	243.05	31.69
		PA	0.24	95.90	11.70	1.04	112.73	13.74	2.48	125.94	18.76	5.63	145.79	22.81
	Top-96	FCFS	19.14	127.44	13.92	24.93	172.65	17.78	28.16	206.97	26.23	29.15	223.41	29.98
		PA	0.25	**78.50**	**10.50**	1.03	**89.65**	**12.38**	2.32	**98.96**	**16.23**	5.06	**114.78**	**19.96**

F.1 Scaling Dynamics and Memory Traffic

Table 7 and Figure 7 comprehensively map the Expert Explosion under continuous batching. As concurrency C and batch size B_{step} scale, the baseline model under standard First-Come-First-Serve (FCFS) routing rapidly saturates the memory bandwidth capacity. For instance, at $C = 128$, $B_{\text{step}} = 32$, the baseline model fetches an average of 242.84 unique experts per step ($\approx 95\%$ of the total pool), driving the MoE kernel latency to a prohibitive 31.66 ms. In contrast, the synergistic combination of the Phase-MoE (Top-96) architecture and the Phase-Aware (PA) scheduler fundamentally bounds this traffic. By aggressively aligning the phase spread ($\Delta\varphi$) via the sliding-window mechanism (See Section 4.2), the PA scheduler restricts HBM fetches to just 114.78 unique experts under the same extreme concurrency, securing a strict upper bound on execution latency.

F.2 Phase Spread Alignment Dynamics ($\Delta\varphi$)

Figure 8 tracks the mean intra-batch phase spread ($\Delta\varphi = \max_{s\in\mathcal{E}} \varphi_s - \min_{m\in\mathcal{E}} \varphi_m$) across varying concurrency levels (C) and execution step batch sizes (B_{step}). Under naive FCFS dispatch, temporal variance remains completely unmitigated, exhibiting a consistently high phase dispersion ($\Delta\varphi \approx$ 19–29) regardless of active concurrency. In contrast, the Phase-Aware sliding-window scheduler leverages the density of the active decode queue $\mathcal{Q}_{\text{dec}}$ to compress $\Delta\varphi$ asymptotically toward zero as concurrency scales. While low concurrency ($C = 16$) forces the scheduler to batch across wider phase gaps to sustain GPU throughput (e.g., $\Delta\varphi = 9.03$ at $B_{\text{step}} = 8$), scaling to production-scale

Table 8: **Continuous Batching Scheduling and Completion Metrics.** Detailed evaluation of sequence Time To Completion (TTC) and scheduling overhead across varying request concurrencies ($C \in \{16, 32, 64, 128\}$) and execution step batch sizes ($B_{\text{step}} \in \{8, 16\}$). *Mean* denotes the average number of forward steps required to fully generate a sequence. *P95* and *P99* reflect the 95th and 99th percentile completion steps, quantifying the tail-end starvation introduced when prioritizing temporal phase alignment over strict First-Come-First-Serve (FCFS) dispatch. *Sched* measures the mean CPU execution overhead (in milliseconds) of the continuous batching dispatch algorithm per step. The results demonstrate that the Phase-Aware (PA) scheduler maintains negligible compute overhead (< 0.1 ms) while successfully bounding tail latency via its dynamic starvation threshold.

			$B_{\text{step}} = 8$				$B_{\text{step}} = 16$			
C	**Model**	**Sched.**	**Mean**	**P95**	**P99**	**Sched**	**Mean**	**P95**	**P99**	**Sched**
16	Baseline	FCFS	88.59	185.08	205.96	0.01	41.57	81.97	85.61	0.01
		PA	66.76	131.43	142.53	0.01	41.57	81.97	85.61	0.01
	Top-128	FCFS	92.89	184.79	211.90	0.00	45.16	84.57	87.38	0.00
		PA	72.00	134.27	140.94	0.01	45.16	84.57	87.38	0.01
	Top-96	FCFS	91.92	178.93	210.97	0.00	44.76	84.32	87.27	0.00
		PA	71.14	139.92	154.32	0.01	44.76	84.32	87.27	0.01
32	Baseline	FCFS	174.78	281.52	337.75	0.01	109.98	234.04	263.91	0.01
		PA	138.15	253.35	303.74	0.02	79.97	151.56	169.94	0.01
	Top-128	FCFS	182.41	292.09	341.20	0.01	114.57	236.24	269.89	0.01
		PA	146.58	271.17	331.22	0.02	89.16	173.13	179.75	0.02
	Top-96	FCFS	183.13	292.58	343.60	0.01	113.29	231.53	266.55	0.01
		PA	144.54	267.68	335.67	0.02	84.79	152.98	177.06	0.01
64	Baseline	FCFS	347.91	506.71	539.93	0.01	222.86	363.67	410.75	0.01
		PA	269.88	464.63	628.40	0.02	162.28	224.74	393.55	0.02
	Top-128	FCFS	362.98	522.76	557.65	0.01	231.25	375.33	411.15	0.01
		PA	296.20	492.52	733.00	0.02	173.16	242.76	421.07	0.02
	Top-96	FCFS	361.51	517.46	558.28	0.01	229.68	370.24	412.41	0.01
		PA	292.00	491.44	655.40	0.02	174.15	308.18	418.38	0.02
128	Baseline	FCFS	703.83	976.75	1009.89	0.03	455.09	656.45	697.37	0.03
		PA	507.17	704.00	972.59	0.05	338.12	590.80	879.06	0.05
	Top-128	FCFS	733.38	1016.35	1053.49	0.03	470.00	673.45	716.50	0.03
		PA	554.73	741.00	1312.49	0.05	354.23	588.60	922.52	0.05
	Top-96	FCFS	729.73	1007.30	1044.83	0.03	467.84	669.85	712.53	0.03
		PA	563.87	903.45	1445.64	0.05	349.54	580.25	703.50	0.05

workloads ($C = 128$) enriches the candidate pool, collapsing the phase spread to $\Delta\varphi = 1.03$ for $B_{\text{step}} = 8$ and $\Delta\varphi = 0.25$ for $B_{\text{step}} = 4$.

F.3 Compute–Memory Queue Dynamics: Phase 0 vs. Phase > 0

A core design of our engine is the disaggregation of the compute-bound Phase 0 queue (handling context prefill and chunked block commits) from the memory-bound Phase > 0 queue (handling iterative decoding). To quickly clear context boundaries, the scheduler grants non-preemptive strict priority to Phase 0 sequences. A natural concern is whether this strict prioritization starves the Phase > 0 decoding queue. Empirically, it does not. Because chunked commits are highly infrequent relative to the high number of iterative denoising steps required per block, Phase 0 compute bursts are sparse. The latency costs of these transitions are fully absorbed into the end-to-end Time To Completion (TTC) metrics, demonstrating that the system rapidly clears Phase 0 bottlenecks and stably returns to the memory-bound execution loop without stalling active generation.

Theoretical Livelock and Concurrency Bounds. A theoretical concern with prioritizing the compute-bound queue ($\mathcal{Q}_{\text{comp}}$) is the risk of livelock: under unconstrained, infinite arrivals, continuous Phase 0 prefill requests could perpetually starve the iterative decoding queue ($\mathcal{Q}_{\text{dec}}$), rendering the starvation threshold τ ineffective.

However, this worst-case scenario is structurally prevented by the system's concurrency mechanisms. In our continuous batching environment (and standard production serving engines), the global active request concurrency C is strictly bounded by KV-cache memory capacity. A new Phase 0 request can only enter the system when an active sequence finishes generation, emits an EOS token, and releases its resources. Because sequence completion strictly requires iterative progression through the memory-bound $\mathcal{Q}_{\text{dec}}$, the arrival of new Phase 0 requests is causally gated by the throughput of Phase > 0. Consequently, $\mathcal{Q}_{\text{comp}}$ cannot infinitely self-sustain; it naturally drains and yields execution back to $\mathcal{Q}_{\text{dec}}$ without requiring complex preemption logic or chunked multiplexing.

F.4 Starvation, Tail Latency, and Completion Dynamics

Deviating from strict FCFS dispatch naturally introduces the risk of tail-end sequence starvation. Table 8 tracks the Time To Completion (TTC)—measured in forward steps—alongside the 95th (P95) and 99th (P99) percentiles to quantify this effect.

First, because the three evaluated checkpoints possess inherent generative differences stemming from their specific training curricula (e.g., variations in average decoding confidence and sequence length), TTC step counts should primarily be compared *within* the same model architecture (FCFS vs. PA) rather than across different checkpoints.

Within a given model, the Phase-Aware scheduler actually *improves* the Mean and P95 completion steps across almost all configurations by efficiently grouping contiguous phases, preventing the thrashing of isolated minority phases. At the extreme tail end (P99), the PA scheduler does exhibit higher step counts at high concurrencies, as minority-phase requests wait in the queue for an optimal alignment window. For example, Phase-MoE (Top-96) at $C = 128, B_{\text{step}} = 8$ shows a P99 of 1445.64 steps under PA scheduling, compared to 1044.83 steps under FCFS. However, this increased step count does not strictly equate to proportionate wall-clock degradation. Because the PA scheduler reduces the underlying MoE kernel execution time by up to 38.5% (Table 2), this per-step acceleration acts as a systemic counterbalance to the accumulated queuing delays at the tail end. Furthermore, the dynamic starvation threshold (See Algorithm 1 and Section 4.2) actively forces minority-phase execution when wait times exceed the tolerance bound. While this does not provide an absolute mathematical guarantee of $t_{\text{wait}} \leq \tau$ (e.g., if multiple requests hit τ simultaneously, the strict B_{step} constraint dictates that some must wait for subsequent forward passes), it serves as a strict priority trigger that successfully mitigates unbounded tail latency in practice.

F.5 Scheduling Computational Overhead

Finally, as reported in Table 8, the computational overhead of the Phase-Aware sliding window algorithm is negligible. Even at the maximum tested concurrency ($C = 128$), the dispatch logic

executes in ≤ 0.05 ms on the CPU, which is entirely masked by the asynchronous GPU execution of the > 10 ms MoE kernels.

G Ablation Studies and Analysis

To isolate the contributions of our proposed architecture, we conduct extensive diagnostic ablations on the Phase-MoE training curriculum, routing constraints, and structural dynamics. Unless otherwise specified, all curriculum and dynamics ablations are evaluated using the Phase-MoE (Top-128) configuration to provide a consistent baseline. To maintain clarity, all downstream generative evaluations are reported on the final Supervised Fine-Tuning (SFT) checkpoints, while the memory traffic metric (Unique Experts U) is extracted from continuous batching simulations under production constraints ($C = 128, B_{\text{step}} = 8$). The consolidated results are presented in Table 9.

Table 9: **Ablation of Phase-MoE Components.** Evaluating the impact of pruning constraints, training curriculum, and routing dynamics. Unless explicitly stated otherwise, curriculum and dynamic ablations utilize the Top-128 configuration. Downstream metrics are measured on the final SFT checkpoints. Unique Experts (U) represent average HBM traffic per forward step under continuous batching ($C = 128, B_{\text{step}} = 8$).

Model Configuration		**Downstream Capabilities (SFT)**				**System**
Variant	**Routing Dynamics**	**ARC-C**	**MMLU**	**GSM8K**	**IFEval**	**Experts** (U)
Baselines & Primary Models						
Baseline (Unconstrained)	Standard Top-K	0.861	0.666	0.588	0.543	204.67
Top-128 (Ours)	Soft $\rightarrow$ Hard + Blur + Center	0.845	0.631	0.553	0.493	112.73
Top-96 (Ours)	Soft $\rightarrow$ Hard + Blur + Center	0.834	0.620	0.524	0.468	89.65
Ablations: Block Size (B) Dynamics						
Top-128 ($B = 16$)	Soft $\rightarrow$ Hard + Blur + Center	0.841	0.628	0.548	0.488	114.20
Top-128 ($B = 64$)	Soft $\rightarrow$ Hard + Blur + Center	0.838	0.625	0.545	0.485	111.45
Ablations: Phase 0 (Prefill) Constraints						
Constrained Phase 0 (Top-128)	Hard constraint applied to $\varphi = 0$	0.764	0.531	0.392	0.365	111.90
Constrained Phase 0 (Top-196)	Hard constraint applied to $\varphi = 0$	0.848	0.639	0.562	0.502	162.35
Ablations: Sparsity Limits						
Top-64 (Extreme Pruning)	Soft $\rightarrow$ Hard + Blur + Center	0.772	0.565	0.385	0.340	62.15
Ablations: Curriculum & Dynamics						
Direct Hard Mask (Top-128)	Static Top-128 (No Soft Stage)	0.720	0.505	0.305	0.283	115.10
Direct Hard Mask (Top-96)	Static Top-96 (No Soft Stage)	0.705	0.452	0.250	0.212	89.85
No Mean-Centering	Soft $\rightarrow$ Hard + Blur (No Center)	0.785	0.550	0.435	0.405	88.50
No Gaussian Blur	Soft $\rightarrow$ Hard (No Blur) + Center	0.825	0.610	0.515	0.455	148.32
Moderate Load Balancing	Soft $\rightarrow$ Hard ($\lambda_{\text{lb}} =$ 2e-3, $\lambda_{\text{spec}} =$ 1e-3)	0.821	0.608	0.518	0.472	116.45
Moderate Specialization	Soft $\rightarrow$ Hard ($\lambda_{\text{lb}} =$ 1e-3, $\lambda_{\text{spec}} =$ 2e-3)	0.844	0.629	0.551	0.490	112.55
Weak Specialization ($\lambda_{\text{lb}} \gg \lambda_{\text{spec}}$)	Soft $\rightarrow$ Hard ($\lambda_{\text{lb}} =$ 2e-3, $\lambda_{\text{spec}} =$ 1e-4)	0.765	0.545	0.420	0.395	128.40
Extreme Specialization ($\lambda_{\text{spec}} \gg \lambda_{\text{lb}}$)	Soft $\rightarrow$ Hard ($\lambda_{\text{lb}} =$ 1e-4, $\lambda_{\text{spec}} =$ 2e-3)	0.740	0.510	0.335	0.315	104.20

G.1 Metrics Formulation

To rigorously quantify routing health, we formally define two internal telemetry metrics utilized in our analysis:

1. Phase Max Violation (`phase_max_vio`): Adapted from auxiliary-loss-free balancing literature (Wang et al., 2024), we compute load balance strictly within the active expert pool of a given denoising phase φ. Let $L_{\varphi,j}$ be the number of tokens assigned to expert j at phase φ, and $\mathcal{C}_\varphi$ be the active support set. The ideal mean load is $\bar{L}_{\varphi,\cdot} = \frac{1}{|\mathcal{C}_\varphi|} \sum_{j \in \mathcal{C}_\varphi} L_{\varphi,j}$. The maximum violation for phase φ is:

$$\text{MaxVio}_\varphi = \max_{j \in \mathcal{C}_\varphi} \frac{|L_{\varphi,j} - \bar{L}_{\varphi,\cdot}|}{\bar{L}_{\varphi,\cdot}} \tag{24}$$

The global `phase_max_vio` metric reported during training is the token-weighted average of MaxVio_φ across all phases and layers.

2. Specialization Intensity (`spec_intensity`): We quantify the degree to which an expert's preference shifts across the denoising trajectory by computing the standard deviation of its bias across

the phase dimension. For the bias matrix $M \in \mathbb{R}^{(B+1)\times N}$:

$$\text{SpecIntensity} = \frac{1}{N}\sum_{j=1}^{N}\sqrt{\frac{1}{B}\sum_{\varphi=1}^{B} M_{\varphi,j}^2} \tag{25}$$

Note that because the bias matrix is double-mean-centered (as proven in Appendix B), the expert-wise mean $\bar{M}_{\cdot,j}$ is strictly zero, simplifying the standard deviation. A higher intensity indicates highly disjoint expert utilization across time.

G.2 Impact of Parallel Block Size (B) and Phase Granularity

The parallel block size B directly dictates the number of discrete denoising phases ($B+1$) the model traverses, which in turn defines the temporal granularity of our Phase-Constrained MoE router. Consistent with the architectural sweet-spot identified in the base LLaDA 2.0 models, we find that $B = 32$ provides the optimal balance. Deviating to $B = 16$ or $B = 64$ introduces slight performance degradations, partially driven by how the router allocates expert pools across time.

Coarse Granularity ($B = 16$): At $B = 16$, the denoising trajectory is compressed into just 17 phases. Consequently, each phase encompasses a much wider variance in mask density. This creates a *bigger box* of expert pools, forcing the router to generalize its subset of K_{hard} experts over a broader range of temporal semantics. This slight loss of temporal sharpness marginally degrades reasoning accuracy (e.g., GSM8K drops from 0.553 to 0.548) and slightly inflates batch-level memory traffic ($U \approx 114.20$) because the broader phase bins limit the scheduler's ability to perfectly align temporally identical sequences.

Fine Granularity ($B = 64$): Conversely, $B = 64$ expands the trajectory to 65 highly granular phases. Ordinarily, slicing the temporal dimension this finely risks severe expert fragmentation, as the router might struggle to allocate sufficient capacity to hyper-specific intermediate phases. However, our architecture avoids catastrophic collapse thanks to the 1D Gaussian blur applied during the Soft Shaping stage (Section 3.2). The blur mathematically binds adjacent, fine-grained phases together, creating a continuous-like, overlapping expert pool that drifts along the denoising trajectory. As a result, the model suffers only a minor drop in generative metrics (IFEval drops to 0.485), while the extreme phase granularity allows the Phase-Aware Scheduler to achieve tight clustering, maintaining a low memory traffic footprint ($U \approx 111.45$).

G.3 Impact of Phase 0 Isolation (Prefill Constraints)

A foundational principle of our compute–memory disaggregation is *Phase 0 Isolation* (Section 3.1). We leave the routing for the initial fully revealed prompt ($\varphi = 0$) completely unconstrained because prefill is strictly compute-bound, whereas active iterative denoising ($\varphi > 0$) is memory-bound. To validate this design choice, we ablate the isolation mechanism by enforcing the hard support mask K_{hard} uniformly across all phases, including Phase 0.

As shown in Table 9, applying the *Top-128 constraint to Phase 0* triggers a severe degradation in downstream performance. Reasoning and knowledge benchmarks collapse significantly (e.g., GSM8K drops from 0.553 to 0.392, and MMLU from 0.631 to 0.531). This occurs because the initial prefill stage requires access to a highly diverse, global pool of experts to encode the rich semantic context of the prompt. Truncating this capacity creates an information bottleneck at the beginning of the generation pipeline, which downstream decoding phases cannot recover from.

To restore generative performance while maintaining a constrained Phase 0, the support pool must be vastly expanded. The *Constrained Phase 0 (Top-196)* variant achieves this, demonstrating comparable (and marginally superior) performance to our primary Top-128 model.

While our primary *Top-128 (with unconstrained Phase 0)* is marginally worse in accuracy than a globally constrained *Top-196*, this trade-off is highly optimal for system throughput. By releasing the constraints on Phase 0, we preserve the necessary contextual capacity during the compute-bound prefill phase. Meanwhile, keeping a tight Top-128 bound on Phase > 0 keeps iterative decoding HBM traffic strictly bounded ($U \approx 112.73$). Conversely, using Top-196 across all phases balloons decoding memory traffic ($U \approx 162.35$), defeating the primary objective of mitigating the Expert Explosion under continuous batching.

G.4 Impact of Pruning Aggressiveness (Support Sparsity Lower Bound)

As demonstrated in Section 5, capping the phase-localized expert pool at $K_{\text{hard}} = 128$ or $K_{\text{hard}} = 96$ yields optimal system throughput without catastrophically degrading the generative priors inherited from the AR base model. However, attempting to aggressively prune the support set further triggers a sharp performance cliff.

The *Top-64* variant exhibits severe semantic collapse, particularly in reasoning-heavy benchmarks requiring complex multi-step planning, such as GSM8K (dropping from 0.588 in the baseline to 0.385) and IFEval instruction following (dropping from 0.543 to 0.340). This establishes an empirical lower bound for viable support sparsity. It indicates that while discrete diffusion models exhibit strong temporal specialization along the denoising trajectory, preserving a minimum threshold of diverse expert capacity per phase remains mathematically essential for accurate syntactic formulation and logical reasoning.

G.5 Soft-to-Hard Curriculum vs. Static Initialization

A natural structural question is whether the complex Stage 1 (Soft Shaping) curriculum is necessary, or if the model could simply be initialized with a rigid Top-k phase mask from step zero. To evaluate this, we initialized the *Direct Hard Mask* variants (both Top-128 and Top-96) by passing a sample of the training corpus through the base AR model and locking the top K experts per phase based on pure activation frequencies prior to training.

Because the base AR checkpoint lacks any temporal diffusion dynamics upon initialization, the correlation between denoising phases and expert activations is naturally near zero. Consequently, enforcing a static hard boundary prematurely locks tokens out of optimal experts precisely as the diffusion process attempts to learn temporal specialization. The router thrashes against these hard boundaries, reflected in a severe degradation of the load balancing metric—`phase_max_vio` surges to $\sim$7.68, compared to 1.14 in the soft-curriculum counterpart. This structural lock-in cripples the model's ability to utilize its parameters effectively, resulting in a broad degradation across all generative metrics (e.g., GSM8K drops to 0.250 in Direct Hard Top-96 vs. 0.524 in Soft-to-Hard Top-96).

G.6 Routing Dynamics: Mean-Centering and Routing Domination

To isolate the theoretical guarantees proposed in Appendix B, we ablate the mathematical constraints applied during Soft Shaping.

Removing the double mean-centering operation (*No Mean-Centering*) removes the zero-sum constraint from the leaky integrator. Because the historical push and pull forces ($\lambda_{\text{lb}}, \lambda_{\text{spec}}$) do not decay, the static temporal bias matrix M grows unboundedly. As training progresses, the magnitude of M rapidly outscales the dynamic $[0, 1]$ gating scores produced by the input tokens.

Telemetry logs confirm this routing domination: the clamping saturation rate—the percentage of bias values forced against the ± 0.4 clipping threshold—rises almost linearly as training advances, eventually hitting $\sim$57.4%, compared to just 0.01% in the properly centered model. This effectively freezes the router's gradient-based learning, rendering it blind to semantic input variations. As a result, downstream knowledge retrieval suffers heavily, with MMLU dropping to 0.550. (Note: This semantic blindness also artificially shrinks the batch-level expert union, dropping U to 88.50, because the router repetitively selects the same static experts regardless of token semantics, destroying representational diversity.)

G.7 Routing Dynamics: Gaussian Blur and Support Overlap

Eliminating the 1D spatial blur ($\mathcal{G}_\sigma$) across the phase dimension (*No Gaussian Blur*) drastically impacts inference hardware efficiency more than raw generative quality. The spatial blur acts as a structural locality constraint, empirically ensuring that adjacent denoising phases (φ and $\varphi \pm 1$) converge toward highly overlapping expert support sets.

Without this blur, adjacent phases learn highly disjoint expert subsets, causing the `spec_intensity` to artificially spike. Consequently, even when the Phase-Aware Scheduler

perfectly clusters contiguous requests during continuous batching ($\Delta\varphi \approx 1.03$), the lack of underlying spatial overlap causes the batch-level expert union to explode. As shown in Table 9, the memory traffic (U) jumps from 112.73 to 148.32 unique experts per step. This ablation explicitly proves that the Gaussian blur is an essential architectural component, ensuring the support overlap necessary for the Phase-Aware Scheduler to safely batch asynchronous sequences.

G.8 Push vs. Pull: Load Balancing and Specialization Trade-offs

In Stage 1 (Soft Shaping), the update dynamics are driven by two competing forces: a load-balancing push (λ_{lb}) and a phase-specialization pull (λ_{spec}). Following standard auxiliary-loss-free balancing practices (Wang et al., 2024), our primary models utilize balanced momentum values ($\lambda_{\text{lb}} = \lambda_{\text{spec}} = 0.001$). Testing moderate imbalances reveals that while slightly prioritizing specialization ($\lambda_{\text{spec}} =$ 2e-3) yields virtually no difference from the baseline, slightly over-indexing on load balancing ($\lambda_{\text{lb}} =$ 2e-3) marginally degrades reasoning (e.g., GSM8K drops to 0.518) because experts specialize less effectively prior to hard pruning. We further ablate extreme imbalances to observe the ultimate tension between utilization and phase-dependence:

1. **Weak Specialization ($\lambda_{\text{lb}} \gg \lambda_{\text{spec}}$):** When load balancing heavily dominates the update step, the router is discouraged from developing natural, data-driven temporal preferences. Consequently, the transition to the hard-pruned stage behaves much like a naive, static hard mask: tokens are forced away from their optimal experts simply to satisfy rigid global load constraints. This suppresses generative performance across all benchmarks. Furthermore, because the experts fail to organically cluster into smooth, overlapping phase pools, the batch-level memory traffic (U) becomes moderately bloated ($U \approx 128$) compared to the well-calibrated baseline.
2. **Extreme Specialization ($\lambda_{\text{spec}} \gg \lambda_{\text{lb}}$):** Conversely, when the specialization pull overpowers load balancing, the model over-indexes on the temporal phase at the expense of token semantics. The router essentially becomes semantically blind, rigidly assigning experts based almost entirely on the current mask density φ while disregarding the underlying linguistic content. This semantic collapse cripples reasoning and knowledge retrieval capabilities, driving GSM8K down to 0.335. While the memory footprint remains constrained by the hard pruning mechanism ($U \approx 104$), the severe loss of representational flexibility renders the configuration unusable.

These findings confirm that phase specialization works best as a structural nudge on top of a well-balanced MoE router, rather than the primary routing objective.

H Broader Impacts

Our work focuses on foundational efficiency optimizations for Mixture-of-Experts block diffusion models. The broader societal and ethical considerations of this research are outlined below.

Environmental Sustainability and Green AI: High-concurrency serving of modern large models incurs massive operational energy costs, predominantly driven by repetitive parameter movement across memory hierarchies (HBM to SRAM). By structurally bounding expert activation subsets per denoising phase, Phase-MoE achieves up to a 56.2% reduction in expert memory fetches and up to 38.5% faster kernel execution. This substantial reduction in memory bus pressure directly translates to lower power dissipation per generated sequence, promoting environmentally sustainable AI deployment and reducing the carbon footprint of production clusters.

Hardware Accessibility and Democratization: As generative models scale, high memory bandwidth demands typically restrict efficient deployment to top-tier enterprise accelerators. By decoupling memory traffic from batch concurrency, our co-design enables higher sustained throughput under tighter hardware constraints, facilitating broader academic and open-source accessibility.

Dual-Use and Downstream LLM Considerations: While our contribution is strictly architectural and computational, accelerating sequence generation inherently amplifies both beneficial and adversarial applications of generative AI. Cheaper, lower-latency inference could potentially be leveraged to scale automated disinformation campaigns, toxic content synthesis, or spam. We emphasize that Phase-MoE is orthogonal to model safety and does not bypass safety alignment pipelines (such as safety fine-tuning or output filtering). We encourage practitioners deploying accelerated diffusion architectures to integrate robust auditing and runtime guardrails to mitigate misuse.