DIFFUSION MOE LLMS FOR MEMORY CONSTRAINED DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) generate multiple tokens in parallel through iterative refinement, offering an alternative to sequential autoregressive decoding. Many recent dLLMs adopt Mixture-of-Experts (MoE) backbones, in which each token activates only a small subset of experts. However, when many tokens are processed simultaneously, their combined routes can span much the expert bank. This is particularly costly on memory-constrained devices, where the full expert bank cannot remain in DRAM and non-resident expert weights must be transferred from flash during inference. We propose dynamic just-in-time expert pruning, a training-free method that restricts routing at each layer and denoising step to a compact expert set selected from the model's current router scores. We further introduce an approach that favors experts selected in the preceding iteration, reducing expert-pool turnover and associated memory traffic. Experiments on DiffusionGemma 26B-A4B and SDAR-30B-A3B show that restricting the selected expert working set to 32 of 128 experts preserves benchmark performance within $\sim 1\%$ of the full baseline while substantially reducing expert-weight demand. Trace-driven simulations estimate up to 5–$9\times$ relative modeled speedup on the evaluated mobile hardware configurations.

1 INTRODUCTION

Diffusion language models (dLLMs) iteratively refine multiple token positions within a block or canvas simultaneously (Inception Labs et al., 2025; Zhu et al., 2025; Cheng et al., 2025; DiffusionGemma Team et al., 2026; Zhu et al., 2026). This multi-token processing increases weight reuse and arithmetic intensity per forward pass, potentially improving accelerator utilization relative to autoregressive decoding. These properties are particularly attractive for on-device inference, where memory capacity is limited and generation often operates at batch size one. At the same time, several recent dLLMs adopt Mixture-of-Experts (MoE) backbones (DiffusionGemma Team et al., 2026; Zhu et al., 2026; 2025; Cheng et al., 2025), for which this repeated multi-token workload introduces a distinct memory-management challenge.

In an MoE backbone, each token is routed to a small top-K subset of E experts. This token-level sparsity reduces routed computation, but does not necessarily imply a small expert working set, as different tokens can activate different experts. During autoregressive generation, each decoding step processes and routes a single token through each MoE layer. Diffusion generation instead routes multiple tokens per forward pass, whose expert selections can span a substantial fraction of the expert bank. Thus, despite sparse per-token routing, the *expert working set*, i.e., the set of distinct expert weights required by a diffusion layer-step, can be much larger.

This distinction is particularly important on memory-constrained devices, where the full expert bank may exceed DRAM capacity. Non-resident expert weights must then be transferred from a slower memory tier, such as flash storage, to fast memory, e.g. DRAM, before execution (Skliar et al., 2025). The cost of an MoE layer then depends not only on the per-token routing sparsity (K), but also on how many distinct experts must be made available for the current diffusion step. However, in diffusion MoEs, even the actively used experts may exceed DRAM capacity, and changes in this set across denoising iterations can trigger additional transfers. The relevant deployment cost is therefore determined by two related quantities: the size of the expert working set at each layer-step and its turnover across denoising iterations.

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.

Iterative denoising also offer an opportunity to reduce these costs. Consecutive iterations refine the same sequence, suggesting that their expert requirements may exhibit exploitable temporal structure. This structure differs across diffusion formulations. In masked diffusion, unresolved positions are progressively replaced by accepted tokens, potentially stabilizing routing between nearby iterations. In uniform diffusion, however, intermediate states may be repeatedly perturbed before convergence, allowing greater variation in routing. We therefore ask two questions: **does each denoising layer-step require the broad expert union selected by native routing**, and: **can temporal reuse reduce the movement of expert weights across iterations?**

To investigate these questions, we introduce **just-in-time expert pruning**, a training-free method that explicitly bounds the expert working set during diffusion inference. At every MoE layer and denoising step, we aggregate the model's native router evidence across the current canvas and select a shared expert working set $P_{t,\ell}$ containing N eligible experts. Each token then performs its standard top-K routing within this working set using the original router scores. The working-set size N therefore provides an explicit quality–memory trade-off and can be chosen according to the device's available expert-memory budget.

We further exploit temporal structure across denoising iterations to reducing expert working-set turnover by introducing previous-pool bias (PPB), which gives a modest preference to experts in the preceding working set while still allowing newly supported experts to enter the pool. This separates two complementary objectives: dynamic pruning limits the number of expert weights that need to be available at a layer-step, while temporal reuse reduces how frequently those weights must be replaced. Together, these mechanisms convert the expert working set from an implicit consequence of token-level routing into an explicit and controllable inference resource.

We evaluate our approach on DiffusionGemma 26B-A4B, (DiffusionGemma Team et al., 2026), which uses uniform diffusion, and SDAR-30B-A3B (Cheng et al., 2025), which uses masked/block diffusion, across reasoning and coding benchmarks. Our main contributions are:

- We introduce dynamic just-in-time expert pruning, a training-free method that bounds the expert working set, together with the PPB extension that reduces turnover across denoising iterations.
- We show that restricting restricting routing to a shared expert working set of 32 out of 128 experts preserves benchmark accuracy with little degradation across both models while substantially reducing expert-weight demand.
- Through trace-driven simulation, we quantify how reductions in expert working-set size and turnover translate into generation latency under three modeled mobile hardware configurations. Replaying measured MATH500 routing traces yields average amortized speedups of $5.8\times$ and $9.1\times$ for DiffusionGemma and SDAR, respectively, under explicit 4-bit weight, cache-capacity, and bandwidth assumptions.

2 Related Work

Diffusion language models. Diffusion and block-diffusion language models have emerged as alternatives to autoregressive text generation (Inception Labs et al., 2025; Zhu et al., 2025; Cheng et al., 2025; DiffusionGemma Team et al., 2026; Zhu et al., 2026). Block-diffusion models generate blocks autoregressively while iteratively denoising tokens within each block (Wu et al., 2025). Causal attention across blocks enables KV caching and coherent generation, while within-block denoising refines multiple tokens in parallel (Ma et al., 2026; Arriola et al., 2025). However, bidirectional attention introduces additional computational costs, particularly with MoE backbones (Wei et al., 2026; Jiang et al., 2026).

MoE inference and deployment. Mixture-of-Experts (MoE) models reduce per-token computation by activating only the top-k experts selected by a router at each decoder layer (Fedus et al., 2022; Jiang et al., 2024; DeepSeek-AI et al., 2025). Despite this computational sparsity, dynamic routing and a large parameter footprint complicate efficient deployment on GPUs (Eliseev & Mazur, 2023; Xue et al., 2025; Kamahori et al., 2025), and on edge decices, where even tighter memory constraints require further optimization (Skliar et al., 2025).

Efficient Diffusion MoE LLMs. Diffusion LLMs exhibit strong temporal-spatial consistency throughout the denoising process (Wei et al., 2026; Chen et al., 2026). TEAM exploits it through decoded-token caching, speculative exploration, and limited activation for cold tokens (Wei et al., 2026). We address a complementary question: how far can the shared expert working set be pruned while preserving denoising quality? TIDE exploits cross-step stability in expert activations to periodically refresh GPU-resident experts using an I/O-aware scheduler (Chen et al., 2026). Because it preserves the original router's predictions and executes misses on the CPU, TIDE is a lossless systems optimization. In contrast, we directly modify MoE routing to reduce the number of distinct active experts to trade a controlled approximation for lower compute and expert-weight traffic.

3 METHOD

3.1 DIFFUSION MOE DECODING AND THE WORKING-SET OBJECTIVE

Consider a diffusion language model that denoises a canvas of length L for T iterations. At denoising step t, MoE layer ℓ, and token position i, the router produces a distribution $p_{t,i,\ell}(e)$ over E experts. Native MoE inference selects the K highest-scoring experts independently for each position,

$$R^{\text{native}}_{t,i,\ell} = \text{TopK}_e \; p_{t,i,\ell}(e), \tag{1}$$

where $R^{\text{native}}_{t,i,\ell}$ is the set of active experts for token i in layer ℓ at denoising step t. This operation is sparse at the level of a single token, but a diffusion iteration evaluates an entire canvas in parallel, so the native expert working set at a layer-step (t, ℓ) is given by the union of all token routes:

$$U_{t,\ell} = \bigcup_{i=1}^{L} R^{\text{native}}_{t,i,\ell}. \tag{2}$$

Even when $K \ll E$, different positions can select different experts, so $|U_{t,\ell}|$ can be large. The same effect recurs across denoising iterations. Thus, token-level sparsity does not by itself bound the expert working set required by an MoE layer-step.

The total size of $U_{t,\ell}$ may be too large to fit in the available fast memory budget for layer ℓ. In this case, not all experts in $U_{t,\ell}$ can remain in memory until the next time step, leading to reduced weight reuse and increased slow memory traffic. Furthermore, even if $U_{t,\ell}$ fits entirely in the available fast memory budget, changes in the expert working-set for layer ℓ between layer-steps incur slow memory traffic.

Our proposed just-in-time expert pruning addresses the former issue by constraining the expert working set, while the temporal selection mechanisms introduced later target the latter by increasing expert reuse between denoising steps.

3.2 JUST-IN-TIME EXPERT PRUNING

As illustrated in Figure 1, our central intervention is to identify, at each layer-step, a bounded expert working set of N eligible experts, $P_{t,\ell} \subseteq \{1, \ldots, E\}$, where $K \leq N \leq E$, from which all canvas positions may route. Every position then performs its usual top-K routing, but only among candidates in $P_{t,\ell}$:

$$\widehat{R}_{t,i,\ell} = \text{TopK}_{e \in P_{t,\ell}} \; p_{t,i,\ell}(e), \qquad \widehat{U}_{t,\ell} = \bigcup_{i=1}^{L} \widehat{R}_{t,i,\ell} \subseteq P_{t,\ell}. \tag{3}$$

Native router scores are left unchanged; experts outside $P_{t,\ell}$ are excluded before top-K selection, with model-specific gate normalization applied to the selected routes. In particular, just-in-time pruning neither retrains nor replaces the learned router, and token-specific routing is preserved within the shared pool. By construction, $|\widehat{U}_{t,\ell}| \leq N$, bounding the number of distinct experts activated at each layer-step.

The pool is recomputed independently for every layer and denoising step, allowing it to adapt across both network depth and the diffusion trajectory. The budget N controls the quality–memory trade-off: $N = E$ recovers native routing, while smaller values reduce the maximum expert working set.

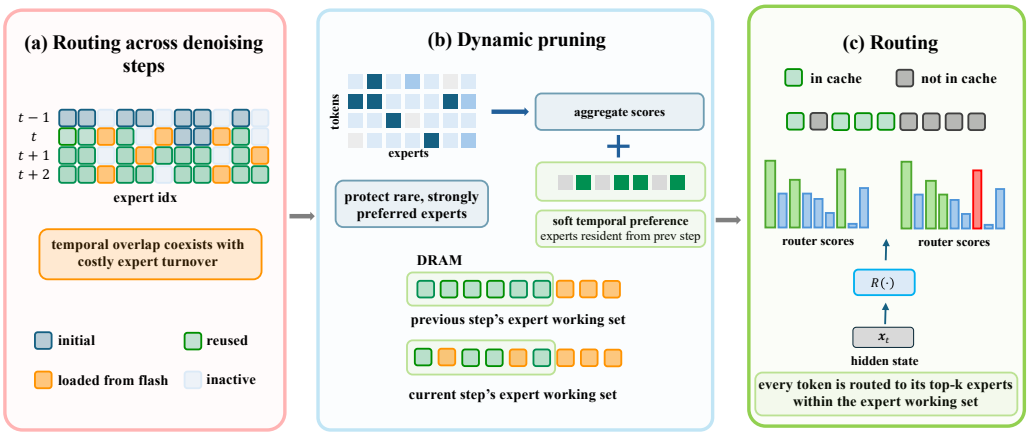


Figure 1: **Overview of our method.** (a) Original token-wise routing exhibits an expert overlap across adjacent denoising steps, yet persistent turnover requires new expert weights to be loaded from flash. (b) At each MoE layer-step, we aggregate current token-level router scores, prune less relevant experts, and softly favor experts selected at the preceding step to construct a compact expert working set of size N. (c) Each token is routed to a subset of the chosen expert working set, retaining its original routing weights. The experts outside the working set are not available for routing.

We therefore treat N as a deployment-controlled memory budget and evaluate the corresponding trade-off between model quality and expert traffic.

3.3 Selecting Experts from Routing Scores

MoE routers naturally provide a notion of expert importance through their routing scores. We aggregate the position-level scores into an expert-wise score $a_{t,\ell}(e)$ and select

$$P_{t,\ell} = \mathrm{TopN}_e \; a_{t,\ell}(e). \tag{4}$$

This selection uses only the current router scores and requires no auxiliary model or calibration data.

Max aggregation. Our primary expert importance score is the maximum router support received by an expert over the current token set,

$$a^{\max}_{t,\ell}(e) = \max_{i \in \{1,\dots,L\}} p_{t,i,\ell}(e). \tag{5}$$

This criterion allows a strong preference from any token to nominate an expert for the shared pool, helping preserve routes that may be important to only a small subset of tokens. We also evaluate sum aggregation that rewards broad average popularity as a natural alternative.

State-balanced max aggregation (SB). In Equation (5), all tokens compete through a single maximum. As denoising progresses, accepted tokens may dominate the canvas, limiting the influence of the remaining unresolved tokens on expert selection. At the same time, accepted tokens remain part of the bidirectional context and may require experts not strongly supported by the unresolved subset alone. To give both groups an independent opportunity to nominate experts, we compute the maximum separately for each state. Let $\mathcal{I}^{\mathrm{m}}_{t-1}$ and $\mathcal{I}^{\mathrm{a}}_{t-1}$ denote the masked/unresolved and accepted token positions from the previous iteration. We compute a separate maximum for each group,

$$a^{g}_{t,\ell}(e) = \max_{i \in \mathcal{I}^{g}_{t-1}} p_{t,i,\ell}(e), \qquad g \in \{\mathrm{m}, \mathrm{a}\}, \tag{6}$$

and form the state-balanced score

$$a^{\mathrm{SB}}_{t,\ell}(e) = a^{\mathrm{m}}_{t,\ell}(e) + a^{\mathrm{a}}_{t,\ell}(e). \tag{7}$$

For SDAR, the two groups are positions remaining masked and positions already accepted when the current forward begins. For DiffusionGemma, we use the acceptance mask produced by the

entropy-bound sampler at the preceding iteration: accepted positions form one group, and positions selected for re-noising form the other. DiffusionGemma recomputes acceptance at each iteration, so accepted status is not necessarily permanent.

This gives both token states a chance to preserve strongly supported experts despite differences in their sizes while keeping the selection causal through the use of previous-iteration state labels.

3.4 Temporal expert reuse for cache-resident pools

Bounding the expert working set size controls it within each layer-step, but the resulting memory traffic also depends on how that pool changes across denoising iterations. Two working sets of equal size can incur very different transfer costs depending on their overlap across iterations. We therefore introduce temporal selection mechanisms that favor experts from the preceding working set when they remain supported by the current router scores.

Previous-pool bias (PPB). Let $a_{t,\ell}(e)$ denote the current expert score and $P_{t-1,\ell}$ the expert pool selected at the preceding denoising step. We bias the current score toward previously selected experts as

$$s_{t,\ell}(e) = a_{t,\ell}(e)\,(1 + \beta\,\mathbf{1}\,[e \in P_{t-1,\ell}])\,, \qquad P_{t,\ell} = \mathrm{TopN}_e\ s_{t,\ell}(e). \tag{8}$$

Here $\beta \geq 0$ controls the strength of temporal reuse, and the first denoising step uses the unmodified score because no previous pool exists. The multiplicative bias does not introduce support for an otherwise irrelevant expert: e.g. if $a_{t,\ell}(e) = 0$, then $s_{t,\ell}(e) = 0$ regardless of its previous membership. For a previously selected expert e_r and a new candidate e_n, the former is preferred whenever $(1+\beta)\,a_{t,\ell}(e_r) \geq a_{t,\ell}(e_n)$. Thus, previously selected experts are retained when their current router support remains sufficiently close to that of competing experts, while strongly supported new experts can still enter the pool. Setting $\beta = 0$ recovers the original independent expert selection.

4 Experiments

4.1 Experimental setup

Models. We evaluate two complementary diffusion MoE architectures: DiffusionGemma 26B-A4B, a uniform-state diffusion model, and SDAR-30B-A3B-Chat, a masked diffusion MoE. For both models, we use the recommended generation settings; more details are provided in Appendix A.

Benchmarks. DiffusionGemma experiments cover the following reasoning benchmarks: AIME2026 (Zhang & Math-AI, 2026), GPQA (Rein et al., 2024), GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), MBPP (Austin et al., 2021), and HumanEval (Chen et al., 2021). SDAR experiments cover GPQA, GSM8K, MATH500, MBPP, and HumanEval using the recommended greedy-decoding protocol. Because SDAR-30B-A3B-Chat-b32 is derived from the Qwen3-30B-A3B-Base checkpoint (Cheng et al., 2025), its performance on advanced mathematical reasoning tasks is limited. On AIME2026, low accuracy combined with the small dataset sizes produces high variance, making comparisons difficult to interpret. We therefore omit AIME evaluations for SDAR. For stochastic decoding settings, we average accuracy over 10 runs for AIME2026 and three runs for all other benchmarks. More details on experimental setup can be found in Appendix A.

Baselines. We compare against native routing and TEAM method (Wei et al., 2026), which exploits temporal and spatial redundancy through decoded-token caching (DCD), limited activation for cold tokens (LAC), and speculative exploration for hot tokens (SEH). We evaluate TEAM using its DCD+LAC configuration and disable the SEH branch, as our study focuses on reducing the expert working set and expert-weight traffic rather than optimizing end-to-end decoding latency. For SDAR, we use the TEAM hyperparameters recommended in the original paper (Wei et al., 2026). Since TEAM was not evaluated on DiffusionGemma, we consider two configurations: one using the paper's recommended hyperparameters and another selected through a grid search to reduce the average number of distinct experts activated per layer-step and align it more closely with our selected working-set size N. The results for TEAM on DiffusionGemma can be found in Appendix C.1.

Metrics. We report zero-shot mean accuracy, the average activated expert count $|\hat{U}_{t,\ell}|$, and the average number of expert cache misses per layer-step. The activated expert count includes only experts that receive at least one token route and can therefore be smaller than the selected working-set size N. A cache miss is a selected working-set expert that is not resident in the layer's DRAM expert cache and must be fetched from flash. To isolate steady-state expert traffic, we exclude the first denoising iteration of each canvas, which incurs an unavoidable cold-start cost to populate the expert cache, and average over all subsequent iterations.

Hardware simulator. We estimate the impact of proposed dynamic expert pruning on token generation speed with a trace-driven memory simulator. The simulator models expert-weight transfers between flash storage and DRAM, together with DRAM cache residency and eviction, to convert each method's expert working set into projected memory traffic and latency. For our simulations, we consider the three hardware configurations listed in Table 1 and make a few realistic assumptions. Specifically, we use a resident model-weight budget of roughly half of the available DRAM, including both non-expert weights and cached expert weights, leaving the remainder for the KV cache, runtime buffers, and the operating system. We assume that the effective DRAM bandwidth and flash read speed are 85% of their peak values to account for contention from other processes. We assume model weights are quantized to a 4-bit datatype. Our analysis targets the memory-constrained regime in which the routed expert bank cannot remain in DRAM and expert execution is limited by weight movement from flash. In this regime, the number of non-resident expert weights requested by each layer-step is a first-order determinant of latency. The simulator is therefore designed to quantify the performance consequence of the resource directly controlled by our method: expert working-set size and turnover.

In our simulated mobile-system experiments, we report relative speedups, comparing all methods under identical hardware, precision, cache-capacity, and bandwidth assumptions, rather than absolute latency since our prototype does not include hardware-specific kernel optimizations. Relative speedups more directly isolate the benefits of reduced expert-weight movement under identical hardware assumptions. We refer to Appendix B for more details.

Table 1: Hardware characteristics of evaluated devices. Apple A18 (Wikipedia, 2026a), Apple A19 Pro (Wikipedia, 2026b) use NVMe storage and Vivo X200T (Wikipedia, 2026c) UFS 4.0 storage.

Setup	NPU	DRAM BW	DRAM size	Flash Read
Apple A18 (iPhone 16)	35 TOPS	60 GB/s	8 GB	3.5 GB/s
Apple A19 Pro (iPhone 17 Pro)	35 TOPS	76.8 GB/s	8 GB	3.5 GB/s
Mediatek Dimensity 9400 (Vivo X200T)	50 TOPS	85 GB/s	16 GB	4.0 GB/s

4.2 RESULTS

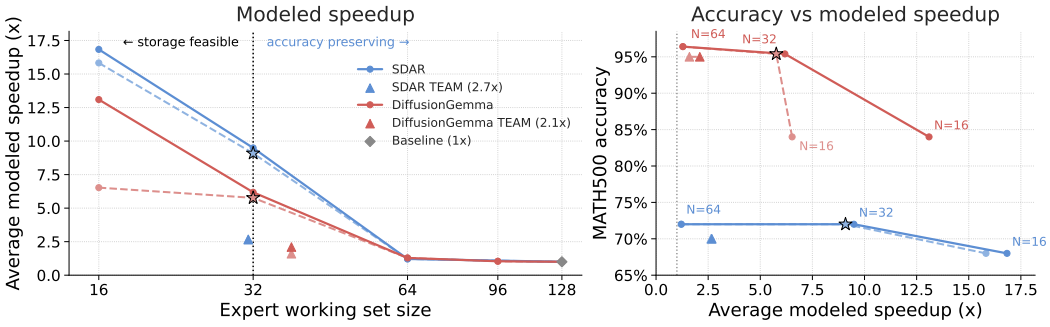


Figure 2: (a) Simulated speedup versus selected expert working-set size N for SDAR-30B-A3B-Chat and DiffusionGemma models measured on MATH500 dataset. Solid lines show average per-denoising step speedup, while dashed lines account for additional canvas steps introduced by expert substitutions. Speedup values are averaged across three hardware setups. Triangles denote TEAM DCD+LAC. Here, we used $\beta = 0.2$ for PPB. (b) Simulated speedup versus accuracy on MATH500 for SDAR-30B-A3B-Chat and DiffusionGemma models. Dashed vertical line indicates Baseline.

Table 2: Comparison of Baseline, TEAM DCD+LAC, and Ours for DiffusionGemma-26B-A4B and SDAR-30B-A3B. Our method uses state-balanced max pruning objective and $N = 32$. Shaded rows report the average number of unique experts activated per layer-step and, when applicable, the reduction relative to the corresponding baseline. HE denotes HumanEval.

	DiffusionGemma-26B-A4B			SDAR-30B-A3B		
Benchmark	Base	TEAM	Ours	Base	TEAM	Ours
AIME26	69.33 ±5.16	67.00 ±5.97	68.67 ±4.22	–	–	–
	65.72	44.80 (-31.8%)	32 (-51.3%)	–	–	–
GPQA	69.70 ±1.27	67.34 ±1.91	68.69 ±1.34	31.31	35.35	37.88
	63.60	44.12 (-30.6%)	32 (-49.7%)	46.87	30.69 (-34.5%)	32 (-31.7%)
GSM8K	94.01 ±0.08	93.73 ±0.23	93.66 ±0.04	89.23	88.70	89.01
	76.48	61.64 (-19.4%)	32 (-58.2%)	58.11	31.80 (-45.3%)	32 (-44.9%)
MATH500	96.13 ±0.50	95.27 ±1.42	96.27 ±0.31	72.00	69.80	72.00
	69.17	52.86 (-23.6%)	32 (-53.7%)	56.53	30.25 (-46.5%)	32 (-43.4%)
MBPP	93.39 ±1.35	92.87 ±0.81	92.22 ±0.67	75.49	73.15	74.71
	72.60	59.94 (-17.4%)	32 (-55.9%)	52.29	28.69 (-45.1%)	32 (-38.8%)
HE	95.73 ±0.00	94.92 ±0.70	95.12 ±0.61	77.44	76.83	79.87
	72.90	56.38 (-22.7%)	32 (-56.1%)	52.47	29.63 (-43.5%)	32 (-39.0%)

Main results. Table 2 shows that our method preserves near-baseline accuracy across both models while restricting each MoE layer-step to at most 32 experts. For DiffusionGemma, this reduces expert activation by 49.7–58.2%. For SDAR, our method reduces expert activation by 31.7–44.9% while matching the baseline on MATH500, and exceeding it on GPQA and HumanEval. We use the recommended TEAM parameters for both DiffusionGemma and SDAR as mentioned in Wei et al. (2026). Since the original TEAM evaluation includes only SDAR, we additionally perform a small parameter sweep for DiffusionGemma to reduce the average number of unique experts activated per layer-step. Appendix C.1 provides additional details and comparisons.

Simulations. For all simulations, we assume that 32 experts fit in DRAM and compute expert misses and turnover accordingly. Figure 2 and Table 3 show a clear efficiency knee at $N = 32$, where the expert pool fits in DRAM and weight traffic drops substantially. At this point, PPB achieves $6.2\times$/$5.8\times$ per-step/amortized speedups for DiffusionGemma and $9.4\times$/$9.1\times$ for SDAR, while retaining baseline-level accuracy. SB max similarly achieves $5.1\times$/$4.7\times$ and $6.0\times$/$5.1\times$ speedups, respectively, while matching or exceeding baseline accuracy. Increasing the pool to $N \geq 64$ yields little accuracy improvement but reduces speedups to 1.1–$1.3\times$, consistent with the flash-bound behavior represented by the model once the pool exceeds DRAM capacity. Because our methods modify the original routing assignments, they may perturb the denoising trajectory and may require additional denoising steps. We account for this overhead in the amortized speedups reported in Table 3, with further details provided in Appendices E.1 and E.2. Although we observe this overhead primarily for DiffusionGemma in practice, the substantial reduction in flash reads preserves large projected gains under the modeled assumptions. Conversely, $N = 16$ provides notably higher nominal speedups but incurs large accuracy losses and, for DiffusionGemma, extra denoising steps that reduce the amortized gain. The consistent trends across hardware profiles identify $N = 32$ as a favorable operating point at the evaluated cache capacity for converting expert locality into substantial speedup without sacrificing with the benchmark-dependent accuracy reported in Tables 2 and 3.

4.3 Ablation study

We ablate the principal design choices of dynamic expert pruning on DiffusionGemma-26B-A4B: the size of the expert working set, the pruning objective, and the use of diffusion-state information.

Size of the expert routing pool. The pool size N directly controls the quality–memory trade-off. Table 4 varies $N \in \{16, 32, 48, 64, 96, 128\}$ while keeping the state-balanced max selection

Table 3: Modeled speedup over original routing for DiffusionGemma and SDAR on MATH500. Apple represents both the A18 and A19 Pro setups, which have same flash read speed. MTK represents MediaTek Dimensity 9400. The two values of N reported for TEAM correspond to DiffusionGemma and SDAR, respectively. For PPB, we have used $\beta = 0.2$

			DiffusionGemma				SDAR			
Method	N	Metric	Accuracy	Apple	MTK	Avg	Accuracy	Apple	MTK	Avg
PPB	16	Per-step	84.0%	12.6	14.0	13.1	68.0%	16.6	17.0	16.8
PPB	16	Amortized	84.0%	6.3	7.0	6.5	68.0%	15.7	16.0	15.8
PPB	32	Per-step	95.4%	6.1	6.4	6.2	72.0%	9.4	9.5	9.4
PPB	32	Amortized	95.4%	5.7	5.9	5.8	72.0%	9.0	9.1	9.1
PPB	64	Per-step	96.4%	1.3	1.3	1.3	71.6%	1.2	1.2	1.2
PPB	64	Amortized	96.4%	1.3	1.3	1.3	71.6%	1.2	1.2	1.2
PPB	96	Per-step	95.8%	1.0	1.0	1.0	72.0%	1.0	1.0	1.0
PPB	96	Amortized	95.8%	1.0	1.0	1.0	72.0%	1.0	1.0	1.0
SB max	16	Per-step	85.5%	12.0	13.2	12.4	70.0%	14.4	14.7	14.5
SB max	16	Amortized	85.5%	5.4	5.9	5.5	70.0%	11.1	11.3	11.2
SB max	32	Per-step	96.3%	5.0	5.2	5.1	72.0%	6.0	6.0	6.0
SB max	32	Amortized	96.3%	4.6	4.8	4.7	72.0%	5.1	5.2	5.1
SB max	64	Per-step	96.3%	1.3	1.3	1.3	72.0%	1.2	1.2	1.2
SB max	64	Amortized	96.3%	1.3	1.3	1.3	72.0%	1.1	1.1	1.1
SB max	96	Per-step	96.1%	1.0	1.0	1.0	72.0%	1.0	1.0	1.0
SB max	96	Amortized	96.1%	1.0	1.0	1.0	72.0%	1.0	1.0	1.0
TEAM DCD+LAC	43 / 31	Per-step	95.3%	2.1	2.1	2.1	70.0%	2.67	2.68	2.7
TEAM DCD+LAC	43 / 31	Amortized	95.3%	1.6	1.6	1.6	70.0%	2.63	2.65	2.6
Baseline	128	Per-step	96.1%	1.0	1.0	1.0	72.0%	1.0	1.0	1.0

objective fixed; same experimental grid for max pruning objective can be found in Appendix D. Restricting the pool to 16 experts is too aggressive and causes a large accuracy loss. Increasing N to 32 experts recovers nearly all of the unconstrained performance: its accuracy is only $\sim 1\%$ point below the $N = 128$ result while retaining one quarter of the expert bank. Increasing $N > 32$ produces only small, non-monotonic differences, indicating that performance has largely saturated.

Table 4: Accuracy for DiffusionGemma using the state-balanced max pruning objective across N.

N	MATH500	GPQA	GSM8K	HumanEval	MBPP
16	85.47 ± 0.64	52.86 ± 1.46	86.32 ± 0.91	92.07 ± 1.06	88.07 ± 0.98
32	96.27 ± 0.31	68.69 ± 1.34	93.66 ± 0.04	95.12 ± 0.61	92.22 ± 0.67
48	96.07 ± 0.42	70.03 ± 0.77	94.01 ± 0.33	95.53 ± 0.35	93.77 ± 0.00
64	96.27 ± 0.42	72.05 ± 1.27	94.04 ± 0.12	95.33 ± 0.70	92.87 ± 0.90
96	96.07 ± 0.31	69.53 ± 0.29	92.87 ± 2.43	95.33 ± 0.70	93.64 ± 0.98
128	96.13 ± 0.50	69.70 ± 1.27	94.01 ± 0.08	95.73 ± 0.00	93.39 ± 1.35

Pruning objective. We next fix the routing pool to $N = 32$ and compare sum and max aggregation. Sum favors experts receiving moderate support from many canvas positions, whereas max allows a single position with a strong routing preference to nominate an expert. As shown in Table 5, max aggregation consistently outperforms sum aggregation. This suggests that preserving rare but strongly preferred routes is more important than maximizing total router mass assigned to the pool.

State-informed pruning. Max pruning favors experts that are highly useful at a few steps, but may remove experts that are consistently useful at other stages of generation as shown in Table 6. By selecting experts with balanced coverage across states, state-balanced max retains this complementary capacity. The effect is most visible on SDAR: it improves GPQA and HumanEval over both baseline routing and standard max pruning, while matching or nearly matching baseline performance on the remaining tasks. On DiffusionGemma, all methods are generally close, but state-balanced max avoids several of the larger regressions of max pruning.

Table 5: Accuracy (%) for DiffusionGemma using sum and max pruning objectives.

Benchmark	Baseline	Sum	Max
AIME2026	69.33 ±5.16	52.22 ±1.92	68.33 ±4.51
MATH500	96.13 ±0.50	92.20 ±0.60	96.10 ±0.26
GPQA	69.70 ±1.27	62.50 ±0.33	69.19 ±1.82
GSM8K	94.01 ±0.08	92.39 ±0.44	93.10 ±0.13
HumanEval	95.73 ±0.00	92.07 ±2.11	93.90 ±0.61
MBPP	93.39 ±1.35	89.75 ±0.81	92.61 ±1.40

Table 6: Comparison of baseline routing, max, and state-balanced (SB) max pruning objectives.

	DiffusionGemma-26B-A4B			SDAR-30B-A3B		
Benchmark	Base	Max	SB Max	Base	Max	SB Max
AIME26	69.33 ±5.16	68.33 ±4.51	68.67 ±4.22	–	–	–
GPQA	69.70 ±1.27	69.19 ±1.82	68.69 ±1.34	31.31	32.83	37.88
GSM8K	94.01 ±0.08	93.10 ±0.13	93.66 ±0.04	89.23	88.86	89.01
MATH500	96.13 ±0.50	96.10 ±0.26	96.27 ±0.31	72.00	70.00	72.00
MBPP	93.39 ±1.35	92.61 ±1.40	92.22 ±0.67	75.49	73.93	74.71
HumanEval	95.73 ±0.00	93.90 ±0.61	95.12 ±0.61	77.44	79.27	79.87

β parameter value. We additionally ablate the effect of the previous-pool bias used in PPB (Equation (8)) by sweeping $\beta \in \{0, 0.1, \ldots, 1.0\}$. Detailed results are provided in Appendix E.1 and Appendix E.2. Alongside accuracy, we report the average number of expert cache misses for PPB and the baseline. As β increases, we observe an approximately linear reduction in expert cache misses and a modest change in accuracy as visualized in Figure 3 in Appendix E.1. At the same time, for DiffusionGemma, denoising overhead grows in a near-linear fashion, indicating a trade-off between cache efficiency and adherence to the original trajectory. Interestingly, SDAR does not exhibit the same behavior. Overhead remains consistently low and, in some cases, the total number of denoising steps even decreases. We hypothesize that this difference stems from the underlying diffusion paradigms. In uniform diffusion, tokens that are not accepted are re-noised at every iteration, whereas masked diffusion keeps them in the `[MASK]` state until they are decoded. As a result, hidden-state evolution is more constrained in masked diffusion, potentially requiring fewer distinct experts and making routing perturbations less disruptive. Generally, $\beta = 0.2$ provides a favorable trade-off between accuracy, expert cache misses, and denoising overhead.

5 Conclusion and Discussion

In this paper, we address deployment of Diffusion MoE LLMs on memory constrained devices. We show that the broad expert working sets induced by original MoE routing are not necessary for effective diffusion MoE inference and deployment. Our training-free expert pruning method makes this working set an explicit, controllable resource, while additional bias-induced PPB method exploits temporal reuse to reduce expert working-set turnover across denoising iterations.

Across DiffusionGemma and SDAR, restricting routing to 32 of 128 experts preserves performance within $\sim 1\%$ and substantially reduces requested expert-weight traffic. Trace-driven simulations map these reductions into $5.8\times$ and $9.1\times$ average speedup on MATH500 for two models, respectively. These results suggest that controlling both the size and turnover of the expert working set is a practical path toward efficient diffusion MoE inference on memory-constrained devices.

Our study has several limitations. First, our trace-driven hardware simulator is a simplified approximation of the evaluated systems and does not capture all aspects of a production-grade software stack. Second, our evaluation of TEAM DCD+LAC does not use its speculative exploration for hot tokens (SEH), because this study focuses on speedups arising from reduced memory traffic rather than acceleration through speculative decoding.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting parts of a research paper and edit a research paper to improve readability. All AI-assisted content was carefully reviewed by the authors through reading and verification. We also used AI tools to support qualitative and thematic data analysis by generating plots and tables from experimental results. We reviewed all generated analysis code and verified that every plot and figure was produced from the correct experimental data. Additionally, we used generative AI tools to create and edit software code used for FLOPs accounting (in Appendix B). We reviewed and cross-checked all generated code and verified the correctness of the results.

We did not use generative AI tools to design or provide feedback on research methodology or experiments, interpret results, propose or refine hypotheses, help develop theoretical models or conceptual frameworks, implement methods, or clean and reformat datasets.

The following usecases were not applicable to this work: Generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and assist with translation.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv [cs.LG]*, March 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv [cs.PL]*, August 2021.

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

Zhiben Chen, Youpeng Zhao, Yang Sui, Jun Wang, and Yuzhang Shang. Tide: Efficient and lossless moe diffusion llm inference with i/o-aware expert offload. *arXiv preprint arXiv:2605.20179*, 2026.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL `https://arxiv.org/abs/2110.14168`.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang,

Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

DiffusionGemma Team, 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 Connor, Sertan Girgin, Maarten Grootendorst, Alon Levkovitch, Eliya Nachmani, Omar Sanseviero, Piotr Stanczyk, Quentin Berthet, Andrew Campbell, Clément Crepy, Valentin De Bortoli, Arnaud Doucet, Romuald Elie, Alexandre Galashov, Klaus Greff, Alexis Jacq, David Ruhe, Yu-Han Wu, Sebastian Flennerhag, Brendan O'Donoghue, George Scrivener, and Shantanu Thakoor. Diffusiongemma technical report, 2026. URL https://arxiv.org/abs/2608.00146.

Artyom Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading, 2023. URL https://arxiv.org/abs/2312.17238.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

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

Haoyun Jiang, Junqi He, Muyi Wang, Fanqin Zeng, Feng Hong, Geng Yu, Pengyi Chen, Yushi Ye, Yuting Cao, Yicheng Fu, Ziyi Tang, Haolin Li, Yuchen Xiong, Zhiyong Chen, Xiaofeng Cao, Xiangtao Li, Bo Han, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. A comparative survey of inference acceleration for DLLMs against AR-LLMs: No free lunch. *Preprints*, May 2026.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N5fVv6PZGz.

Xiangyu Ma, Teng Xiao, Zuchao Li, and Lefei Zhang. From AR to diffusion: Efficiently adapting large language models with strictly causal and elastic horizons. *arXiv [cs.CL]*, April 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Andrii Skliar, Ties van Rozendaal, Romain Lepert, Todor Boinovski, Mart Van Baalen, Markus Nagel, Paul N. Whatmough, and Babak Ehteshami Bejnordi. Mixture of cache-conditional experts for efficient mobile device inference. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=ul4W26KEKz.

Linye Wei, Zixiang Luo, Pingzhi Tang, and Meng Li. Team: Temporal-spatial consistency guided expert activation for moe diffusion language model acceleration, 2026. URL https://arxiv.org/abs/2602.08404.

Wikipedia. Apple a18. https://en.wikipedia.org/wiki/Apple_A18, 2026a.

Wikipedia. Apple a19. https://en.wikipedia.org/wiki/Apple_A19, 2026b.

Wikipedia. List of mediatek systems on chips. https://en.wikipedia.org/wiki/List_of_MediaTek_systems_on_chips, 2026c.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv [cs.CL]*, May 2025.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. Moe-infinity: Efficient moe inference on personal machines with sparsity-aware expert cache, 2025. URL https://arxiv.org/abs/2401.14361.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

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

Fengqi Zhu, Shaoxuan Xu, Jingyang Ou, Zebin You, Yipeng Xing, Huabin Liu, Xiaolu Zhang, Jun Zhou, Zhenzhong Lan, Yankai Lin, Wayne Xin Zhao, Jianguo Li, Chongxuan Li, and Ji-Rong Wen. Llada moe v2: Scaling mixture-of-experts diffusion language models, 2026. URL https://arxiv.org/abs/2608.03457.

A IMPLEMENTATION DETAILS

DiffusionGemma 26B-A4B-it is a uniform-state diffusion model with 128 experts, top-8 expert routing, 30 MoE layers, and a 256-token canvas. We use its default generation configuration: at most 48 denoising steps, entropy-bound token acceptance with a bound of 0.1, a linear temperature schedule from 0.8 at the first denoising step to 0.4 at the last, and adaptive stopping when the predicted canvas remains unchanged for one step and the mean token entropy falls below 0.005. SDAR-30B-A3B-Chat-b32 is a masked diffusion MoE with 128 experts, top-8 expert routing, 48 layers, and a 32-token diffusion block. We denoise each block for 32 steps using greedy decoding (temperature 1.0, sampling top-k of 1, and top-p of 1.0), static low-confidence remasking, and a configured confidence threshold of 0.90.

We set the generation budget to 32,768 tokens for AIME2026, MATH500, and GPQA Diamond; 16,384 tokens for GSM8K; and 4,096 tokens for HumanEval and MBPP. DiffusionGemma's reasoning mode is enabled for AIME2026, MATH500, and GPQA Diamond and disabled for the remaining benchmarks.

B SIMULATION METHOLOGY

Our evaluation combines observed routing traces with an analytical memory-hierarchy model. Accuracy, layer-wise expert selections, and denoising-step counts are collected from complete benchmark inference runs, including additional iterations caused by routing changes. Replaying these traces through the modeled cache yields estimates of cache misses, expert-weight traffic, and latency.

All methods use identical compute, memory, quantization, and cache assumptions, with ideal overlap between expert computation and weight delivery. Relative modeled speedups quantify differences in expert demand and denoising work under these assumptions. The reported speedups characterize the deployment scenarios defined by these assumptions; validation against measured device latency remains future work.

To measure speedups, we replay each method's routing trace through a per-layer LRU expert cache, sized according to the hardware memory budget. At every post-initial denoising layer-step, we compute the cache misses from the working expert set demand. We convert these counts into memory traffic using the quantized size of each expert: resident weights are loaded from DRAM, whereas each cache miss incurs a fetch from flash and updates the cache state. Let r_{miss} denote the cache-miss rate, D the number of requested experts, P_e the parameters per expert, and s_w the storage size of each quantized parameter. Then $B_{\text{flash}} = r_{\text{miss}} D P_e s_w$ and $B_{\text{DRAM}} = (1 - r_{\text{miss}}) D P_e s_w$.

We separately estimate the FLOPs of the active top-k expert computations and the modeled non-MoE denoiser components. We estimate MoE latency as $\max(F/\Theta_{\text{compute}}, B_{\text{DRAM}}/\beta_{\text{DRAM}}, B_{\text{flash}}/\beta_{\text{flash}})$ and add the sequential non-MoE component times. Here, F denotes FLOPs, B_{DRAM} and B_{flash} denote bytes transferred from DRAM and flash, and Θ_{compute}, β_{DRAM}, and β_{flash} denote compute throughput and DRAM and flash bandwidths, respectively. This formulation follows a roofline-style execution model in which expert computation, DRAM weight delivery, and flash weight delivery determine the critical-path latency. Taking their maximum represents ideal overlap among these components. We then add the sequential latencies of the modeled non-MoE components. Because the same overlap assumption is applied to all methods, relative differences arise from their measured expert demand and compute traces.

We report relative speedup as $T_{\text{native}}/T_{\text{method}}$, comparing all methods under identical hardware, precision, cache-capacity, and bandwidth assumptions.

B.1 SDAR FLOP ACCOUNTING

SDAR has hidden dimension $d = 2048$, expert intermediate dimension $d_e = 768$, $k = 8$ routed experts per token, $L = 48$ decoder layers, and a denoising block of $Q = 32$ tokens. Each expert contains

$$P_e = 3dd_e \tag{9}$$

parameters, corresponding to the gate, up, and down projections. The sparse-MoE FLOPs for one layer-step are therefore

$$F_{\mathrm{MoE}} = 2QkP_e. \tag{10}$$

For SDAR, this gives $P_e = 4{,}718{,}592$ parameters and $F_{\mathrm{MoE}} = 2.416$ GFLOPs per layer-step.

SDAR uses $n_h = 32$ query heads, $n_{\mathrm{kv}} = 4$ key–value heads, head dimension $d_h = 128$, and context length $C = 4096$. The attention parameter count used by the simulator is

$$P_{\mathrm{attn}} = 2dn_h d_h + 2dn_{\mathrm{kv}} d_h + 2d_h. \tag{11}$$

The attention FLOPs per layer-step are

$$F_{\mathrm{attn}} = 2\left(QP_{\mathrm{attn}} + 2n_h QCd_h\right), \tag{12}$$

where the first term accounts for the attention projections and the second accounts for the query–key and attention–value products. This gives $F_{\mathrm{attn}} = 3.355$ GFLOPs per layer-step. The total modeled FLOPs for one SDAR denoising forward are

$$F_{\mathrm{SDAR}} = L\left(F_{\mathrm{MoE}} + F_{\mathrm{attn}}\right) = 277.0 \text{ GFLOPs}. \tag{13}$$

B.2 DiffusionGemma FLOP Accounting

DiffusionGemma has hidden dimension $d = 2816$, expert intermediate dimension $d_e = 704$, dense intermediate dimension $d_{\mathrm{dense}} = 2112$, $k = 8$ routed experts per token, $L = 30$ decoder layers, and a canvas of $Q = 256$ tokens. The expert and dense-MLP parameter counts are

$$P_e = 3dd_e, \qquad P_{\mathrm{dense}} = 3dd_{\mathrm{dense}}. \tag{14}$$

The corresponding per-layer FLOPs are

$$F_{\mathrm{MoE}} = 2QkP_e, \qquad F_{\mathrm{dense}} = 2QP_{\mathrm{dense}}. \tag{15}$$

Numerically, these are 24.361 and 9.135 GFLOPs per layer, respectively.

DiffusionGemma contains $L_s = 25$ sliding-attention layers and $L_f = 5$ full-attention layers. For attention type $x \in \{s, f\}$, let $d_h^{(x)}$ be its head dimension, $n_{\mathrm{kv}}^{(x)}$ its number of key–value heads, and

$$q_x = n_h d_h^{(x)}, \qquad u_x = n_{\mathrm{kv}}^{(x)} d_h^{(x)}, \tag{16}$$

where $n_h = 16$. The attention projection parameter count is

$$P_{\mathrm{attn}}^{(x)} = dq_x + du_x + \delta_x du_x + q_x d, \tag{17}$$

where $\delta_s = 1$ because sliding attention uses a separate value projection, while $\delta_f = 0$ because full attention reuses the key representation as the value representation. The attended sequence lengths are

$$K_s = \min(C, W_s) + Q, \qquad K_f = C + Q, \tag{18}$$

where $C = 4096$ and $W_s = 1024$. The FLOPs of one attention layer of type x are

$$F_{\mathrm{attn}}^{(x)} = 2\left(QP_{\mathrm{attn}}^{(x)} + 2n_h QK_x d_h^{(x)}\right). \tag{19}$$

We use $(n_{\mathrm{kv}}^{(s)}, d_h^{(s)}) = (8, 256)$ and $(n_{\mathrm{kv}}^{(f)}, d_h^{(f)}) = (2, 512)$.

For each post-initial denoising forward, the simulator additionally includes self-conditioning, the language-model head, and soft-embedding computation. Let $V = 262{,}144$ be the vocabulary size and $P_{\mathrm{vocab}} = dV$. Their combined FLOPs are

$$F_{\mathrm{fixed}} = 2Q\left(P_{\mathrm{dense}} + 2P_{\mathrm{vocab}}\right). \tag{20}$$

The total modeled FLOPs for one post-initial DiffusionGemma denoising forward are therefore

$$\begin{aligned} F_{\mathrm{DiffusionGemma}} &= LF_{\mathrm{MoE}} + LF_{\mathrm{dense}} + L_s F_{\mathrm{attn}}^{(s)} + L_f F_{\mathrm{attn}}^{(f)} + F_{\mathrm{fixed}} \\ &= 2.655 \text{ TFLOPs}. \end{aligned} \tag{21}$$

C TEAM EXPERIMENTS

C.1 TEAM SWEEP

We tune TEAM using a structured six-point screening procedure over its cold-confidence threshold (τ), distance threshold (d), and number of protected hot tokens (h). Starting from the untuned upstream configuration $(\tau, d, h) = (0.70, 3, 2)$, we evaluate $\tau \in \{0.95, 0.98\}$ in combination with increasingly restrictive settings $(d, h) \in \{(1, 1), (0, 1), (0, 0)\}$. We report every screened configuration together with its accuracy and average number of activated experts.

Configuration selection is performed independently of our method. Among settings whose accuracy falls within one percentage point of the highest observed TEAM accuracy, we select the configuration that activates the fewest experts, breaking ties by accuracy. The selected configurations are subsequently evaluated using additional random seeds. The results are reported in Table 7 and Table 8.

Table 7: TEAM parameter ablation on DiffusionGemma-26B-A4B for reasoning benchmarks. The setting is (τ, d, h): cold-confidence threshold, distance threshold, and number of protected hot tokens. Each entry reports accuracy (%) / average unique experts per layer-step. Bold denotes selected setups. We choose the setting with the number of unique experts closest to 32. For similar outcomes, we select the option with higher accuracy.

Setting	AIME26	GPQA	GSM8K	MATH500
$(0.70, 3, 2)$	67.00 / 44.80	67.34 / 44.12	93.73 / 61.64	95.27 / 52.86
$(0.95, 1, 1)$	46.67 / 40.76	68.18 / 36.19	93.71 / 57.84	95.80 / 43.75
$(0.98, 1, 1)$	63.33 / 30.19	67.68 / 37.81	93.86 / 56.20	95.80 / 39.22
$(0.95, 0, 1)$	**63.33 / 31.52**	69.19 / 29.41	93.56 / 56.86	95.27 / 39.05
$(0.98, 0, 1)$	53.33 / 27.98	67.68 / 27.88	**94.16 / 54.37**	**95.40 / 35.90**
$(0.95, 0, 0)$	70.00 / 38.20	66.67 / 33.50	93.86 / 56.71	94.20 / 34.84
$(0.98, 0, 0)$	70.00 / 28.66	**68.18 / 30.23**	93.71 / 54.03	93.09 / 43.53

Table 8: TEAM parameter ablation on DiffusionGemma-26B-A4B for coding benchmarks. The setting is (τ, d, h): cold-confidence threshold, distance threshold, and number of protected hot tokens. Each entry reports accuracy (%) / average unique experts per layer-step. Bold denotes selected setups.

Setting	MBPP	HumanEval
$(0.70, 3, 2)$	92.87 / 59.94	94.92 / 56.38
$(0.95, 1, 1)$	92.61 / 49.92	93.29 / 51.69
$(0.98, 1, 1)$	91.83 / 48.37	93.90 / 50.56
$(0.95, 0, 1)$	91.44 / 47.96	95.12 / 49.52
$(0.98, 0, 1)$	93.00 / 43.41	95.12 / 46.05
$(0.95, 0, 0)$	93.00 / 47.09	95.73 / 48.59
$(0.98, 0, 0)$	**93.00 / 42.68**	**94.51 / 45.56**

C.2 EVALUATIONS

In Table 9, we compare the Baseline, a tuned TEAM configuration selected from a set of aggressively compressed operating points, and our method on DiffusionGemma-26B-A4B. Within the evaluated parameter grid, TEAM exhibits a different accuracy–expert-demand trade-off across the two model families. In contrast, our experiments on SDAR indicate a more favorable tradeoff between the number of active experts and accuracy, suggesting that TEAM is better suited to the routing characteristics of SDAR than those of DiffusionGemma.

Table 9: Comparison of Baseline, tuned TEAM, and Ours for DiffusionGemma-26B-A4B. Accuracy is reported as mean ± standard deviation. Our method uses state-balanced max pruning objective and $N = 32$. Results are averaged over 10 seeds for AIME2025 and AIME2026, and over 3 seeds for all other benchmarks. Shaded rows report the average number of unique experts activated per layer-step and, when applicable, the reduction relative to the corresponding baseline.

Benchmark	Base	TEAM	Ours
AIME25	66.67 ±4.71	64.00 ±3.44	70.33 ±5.08
	64.96	33 (-49.2%)	32 (-50.7%)
AIME26	69.33 ±5.16	65.19 ±5.30	68.67 ±4.22
	65.72	32 (-51.3%)	32 (-51.3%)
GPQA	69.70 ±1.27	67.00 ±2.33	68.69 ±1.34
	63.60	30 (-52.8%)	32 (-49.7%)
GSM8K	94.01 ±0.08	94.29 ±0.16	93.66 ±0.04
	76.48	54 (-29.4%)	32 (-58.2%)
MATH500	96.13 ±0.50	95.27 ±1.42	96.27 ±0.31
	69.17	36 (-48.0%)	32 (-53.7%)
MBPP	93.39 ±1.35	92.74 ±1.19	92.22 ±0.67
	72.60	43 (-40.8%)	32 (-55.9%)
HumanEval	95.73 ±0.00	95.12 ±0.61	95.12 ±0.61
	72.90	46 (-36.9%)	32 (-56.1%)

D Ablation for working expert set size

The max pruning objective exhibits a similar quality–memory trade-off to the state-balanced max objective discussed in the main text. As shown in Table 10, restricting the working set to $N = 16$ is again too aggressive, resulting in substantial accuracy degradation across all benchmarks. Increasing the pool to $N = 32$ recovers most of the unconstrained performance. Averaged across the five benchmarks, the $N = 32$ configuration is only 0.81 percentage points below the $N = 128$ configuration. Beyond $N = 32$, accuracy differences are small and non-monotonic, indicating that performance has largely saturated. However, comparison with Table 4 shows that state balancing considerably improves robustness at the most restrictive setting, $N = 16$, while the two objectives perform comparably once the working expert set is sufficiently large.

Table 10: Accuracy for DiffusionGemma using the max pruning objective across active expert pool sizes. Results are sample means ± sample standard deviations over three random seeds.

N	MATH500	GPQA	GSM8K	HumanEval	MBPP
16	74.73 ± 2.34	48.82 ± 3.25	78.47 ± 0.62	87.40 ± 0.93	82.75 ± 1.25
32	96.10 ± 0.26	69.19 ± 1.82	93.10 ± 0.13	93.90 ± 0.61	92.61 ± 1.40
48	96.20 ± 1.00	70.71 ± 2.67	93.96 ± 0.12	95.12 ± 1.22	93.39 ± 0.78
64	96.80 ± 0.40	70.54 ± 2.04	94.01 ± 0.08	96.14 ± 0.35	94.03 ± 1.25
96	96.73 ± 0.46	69.53 ± 1.17	94.26 ± 0.18	96.14 ± 1.27	93.26 ± 0.59
128	96.13 ± 0.50	69.70 ± 1.27	94.01 ± 0.08	95.73 ± 0.00	93.39 ± 1.35

E Sweep over β value

E.1 DiffusionGemma

Tables 11 and 12 report the effect of biasing expert-pool selection toward experts selected at the preceding denoising step. Even without an explicit previous-pool bias $\beta = 0$ (which collapses to regular state-balanced max pruning objective), PPB substantially reduces the average number

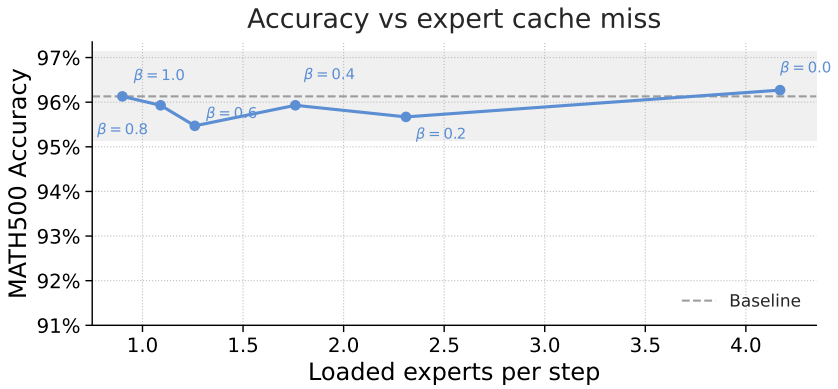


Figure 3: MATH500 accuracy versus average expert cache misses per layer-step for PPB with state-balanced-max routing. The native baseline achieves 96.13% accuracy with 65.94 cache misses per layer-step; its traffic point is omitted for readability. The shaded band indicates $\pm 1\%$ point from baseline accuracy.

of expert cache misses relative to the original routing. Increasing β further reduces cache misses consistently across all benchmarks. For example, on MATH500, cache misses decrease from 4.17 at $\beta = 0$ to 2.31 at $\beta = 0.2$ and 0.90 at $\beta = 1.0$, compared with 65.94 for the baseline model. Similar trends hold for AIME2026, GPQA, GSM8K, MBPP, and HumanEval, where the cache-miss count approaches approximately one expert cache miss per layer-step at the largest bias values. Accuracy is comparatively stable at small and moderate values of β, although aggressive biasing can degrade performance on some benchmarks, At $\beta = 0.2$, accuracy remains close to the native baseline across all tasks, while providing decrease in expert cache misses. In contrast, larger values occasionally cause more pronounced degradation, particularly on AIME2026 and GPQA. The denoising overhead also generally increases with β, although the trend is noisy across individual benchmarks. These results support using $\beta = 0.2$ as a conservative operating point that captures most of the cache-miss reduction while limiting both accuracy degradation and deviation from the native denoising trajectory.

Table 11: Accuracy, cache misses, and overhead for DiffusionGemma across different values of the bias strength β for AIME2026, GPQA, and GSM8K. Acc denotes accuracy, CM denotes average expert cache misses per layer-step, and Base denotes the original model.

	AIME2026			GPQA			GSM8K		
β	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)
Base	69.33	64.45	0.0	69.70	60.24	0.0	94.01	78.07	0.0
0.0	68.67	3.90	+2.4	68.69	4.40	+6.3	93.66	4.11	+6.9
0.1	71.33	2.71	+10.2	67.68	2.45	+11.1	93.28	3.65	+7.7
0.2	69.00	2.12	+7.5	67.00	1.86	+10.7	93.51	2.95	+8.5
0.3	69.33	1.75	+9.3	67.34	1.47	+12.9	93.68	2.51	+7.4
0.4	69.00	1.49	+8.6	67.93	1.21	+13.6	93.33	2.20	+9.1
0.5	66.33	1.27	+15.1	66.33	1.08	+13.1	93.35	1.95	+9.7
0.6	70.33	1.10	+14.4	67.85	0.86	+14.3	93.25	1.75	+11.0
0.7	68.33	1.03	+12.0	65.32	0.96	+19.8	93.53	1.57	+13.6
0.8	68.00	0.84	+26.6	68.01	0.73	+17.1	93.56	1.43	+15.5
0.9	67.41	0.82	+18.7	66.33	0.69	+17.6	93.48	1.31	+17.1
1.0	64.67	0.76	+23.6	65.49	0.56	+21.4	93.30	1.20	+17.0

E.2 SDAR-30B-A3B-Chat-b32

Tables 13 and 14 present the corresponding sweep for SDAR. As with DiffusionGemma, PPB produces a large reduction in expert cache misses even at $\beta = 0$, and increasing β reduces them further

Table 12: Accuracy, cache misses, and overhead for DiffusionGemma across different values of the bias strength β for MATH500, MBPP, and HumanEval. Acc denotes accuracy, CM denotes average expert cache misses per layer-step, and Base denotes the original model.

	MATH500			MBPP			HumanEval		
β	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)
Base	96.13	65.94	0.0	93.39	45.89	0.0	95.73	43.42	0.0
0.0	96.27	4.17	+9.0	92.22	4.94	+2.4	95.12	4.73	+9.85
0.1	96.40	2.86	+9.8	93.39	3.42	+15.1	94.72	3.29	+15.6
0.2	95.4	2.31	+9.4	93.13	2.97	+10.9	94.51	2.97	+9.69
0.3	95.73	1.99	+9.7	93.00	2.59	+7.4	94.51	2.43	+10.2
0.4	95.93	1.76	+10.2	92.74	2.19	+11.9	94.72	2.23	+20.9
0.5	96.00	1.38	+11.6	93.13	2.10	+6.8	94.51	1.75	+21.9
0.6	95.47	1.26	+13.5	93.13	1.77	+17.9	95.33	1.81	+27.5
0.7	95.33	1.15	+14.6	93.13	1.44	+18.6	93.29	1.66	+27.9
0.8	95.93	1.09	+16.7	91.83	1.43	+21.6	93.90	1.53	+30.3
0.9	95.40	0.97	+17.0	93.64	1.27	+31.8	94.92	1.37	+18.5
1.0	96.13	0.90	+18.6	92.61	1.07	+33.9	95.53	1.16	+35.8

on every benchmark. Similarly, most of the available reduction is already obtained using a previous-pool bias.

Unlike DiffusionGemma, SDAR does not exhibit a systematic increase in denoising overhead as β grows. Across the sweep, the overhead remains within approximately $\pm 10\%$ and is frequently negative, indicating that the perturbed routing trajectory can require fewer canvas denoising steps than native routing. For example, the overhead at $\beta = 0.2$ is $+4.43\%$ on MATH500, -7.31% on GPQA, 0.00% on GSM8K and MBPP, and $+3.46\%$ on HumanEval. Accuracy is also relatively insensitive to the bias on MATH500 and GSM8K, while GPQA exhibits substantial run-to-run or configuration-level variation and MBPP shows a modest reduction relative to the native baseline. At $\beta = 0.2$, performance remains close to the baseline overall while cache misses are reduced by roughly one order of magnitude. We therefore retain $\beta = 0.2$ as the default configuration, as larger values provide progressively smaller cache-efficiency gains without consistently improving accuracy or denoising overhead.

Table 13: Accuracy, cache misses, and overhead for SDAR across different values of the bias strength β on MATH500, GPQA, and GSM8K. Acc denotes accuracy, CM denotes average expert cache misses per layer-step, and Base denotes the original model.

	MATH500			GPQA			GSM8K		
β	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)
Base	72.00	22.30	0.00	31.31	22.46	0.00	89.23	27.88	0.00
0.0	72.00	4.31	+0.8	37.88	3.70	+8.1	89.01	4.16	−7.0
0.1	72.00	2.79	+3.95	36.36	2.70	−3.72	88.55	3.04	−3.52
0.2	72.00	2.43	+4.43	33.11	2.30	−7.31	89.01	2.54	+0.00
0.3	72.00	2.12	−3.16	33.33	2.05	−4.22	89.01	2.26	−3.52
0.4	72.00	1.97	−0.65	33.33	1.90	+0.71	90.37	2.08	−1.76
0.5	72.00	1.84	−5.23	36.36	1.79	+3.03	89.01	1.93	+0.00
0.6	72.00	1.65	−3.79	36.87	1.73	−1.20	89.01	1.84	−5.28
0.7	72.00	1.66	+5.88	34.34	1.64	−0.03	89.01	1.76	−3.52
0.8	72.00	1.58	+5.88	36.36	1.60	−8.69	89.01	1.70	−8.80
0.9	72.00	1.53	−4.74	40.91	1.53	−9.74	89.01	1.65	−5.28
1.0	72.00	1.51	−8.49	34.85	1.49	+1.22	89.01	1.58	−7.04

Table 14: Accuracy, cache misses, and overhead for SDAR across different values of the bias strength β on MBPP and HumanEval. Acc denotes accuracy, CM denotes average expert cache misses per layer-step, and Base denotes the original model.

	MBPP			HumanEval		
β	Acc (%)	CM	Overhead (%)	Acc (%)	CM	Overhead (%)
Base	75.49	23.86	0.00	77.44	21.93	0.00
0.0	74.71	4.95	0.0	79.87	4.98	-3.5
0.1	72.37	3.87	$+2.3$	76.22	3.80	$+2.9$
0.2	74.32	3.33	$+2.3$	76.22	3.25	$+3.46$
0.3	73.54	3.08	$+1.1$	76.83	2.86	$+4.3$
0.4	72.37	2.85	$+0.0$	77.44	2.64	$+2.9$
0.5	71.60	2.67	$+1.1$	77.44	2.49	$+1.4$
0.6	71.60	2.51	$+2.3$	76.22	2.32	$+1.4$
0.7	73.15	2.40	$+2.3$	76.22	2.25	$+4.3$
0.8	73.54	2.32	$+2.3$	76.83	2.16	-3.46
0.9	72.76	2.22	$+3.4$	76.22	2.09	$+0.00$
1.0	74.32	2.17	$+2.3$	73.78	1.98	$+2.9$