SEP: SPECULATIVE EXPERT PREFETCHING FOR LOW-LATENCY MIXTURE-OF-EXPERTS INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-experts (MoE) decode is memory-bandwidth bound: each token generation step must stream expert weight matrices from HBM before the expert GEMM can execute. We introduce Speculative Expert Prefetching (SEP), which predicts each token's active experts from the pre-attention hidden state and prefetches expert weights overlapped with attention, routing, and multi-GPU communication. SEP rests on a hardware-algorithm co-design insight, that on-package cache holds more experts than the router's per-GPU top-k, enabling overprovisioned prefetching that relaxes the prediction objective from exact expert prediction to per-GPU coverage. When this overprovisioning is large, spare cache slots tolerate mispredictions and a lightweight predictor suffices, while tight cache capacity demands higher accuracy. A second limit is temporal, as the predictor and the expert weight transfer must together complete before the expert GEMM begins. We evaluate 18 predictor architectures across four modern MoE models, demonstrating strong out-of-distribution generalization across diverse task categories. Our best predictor achieves 98.5–100% accuracy in predicting experts, translating to 1.3–$1.8\times$ decode-GEMM speedups at batch 1 on production MoE kernels.

1 INTRODUCTION

Interactive and agentic LLMs are dominated by *decode*, which generates one token at a time under strict latency constraints. Frontier models are overwhelmingly mixture-of-experts (MoE) (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024; DeepSeek-AI, 2024; Yang et al., 2025; OpenAI, 2025), routing each token to a small subset of experts, making decode memory-bandwidth bound: each step streams large expert weight matrices from HBM, stalling the expert GEMM.

Prefetching offers a direct solution by issuing asynchronous transfers of the predicted experts' weights from HBM into on-package cache (L2/LLC) *before* the expert GEMM issues, overlapping the fetch with work already on the critical path (attention, normalization, routing, and multi-GPU communication). The fundamental challenge with prefetching is that the router reveals which experts are required only immediately before the MoE step, leaving insufficient time to hide the transfer latency. We address this by *predicting* the active experts from the layer's pre-attention hidden state and issuing speculative prefetches early (Figure 1). This formulates speculative expert prefetching as a joint problem: **how to predict** which experts a token will activate, and **how to prefetch** their weights before the expert GEMM begins. Prior expert prefetching work targets single-GPU offload setups where experts are loaded from CPU DRAM or SSD into GPU HBM, a fundamentally different memory hierarchy (Eliseev & Mazur, 2023; Hwang et al., 2024; Yi et al., 2023; Zhong et al., 2024; Yu et al., 2024; Zhu et al., 2025). A further requirement, overlooked in prior work, is that any practical predictor must generalize across workloads, as production serving systems encounter arbitrary input queries, making out-of-distribution (OOD) generalization a first-class optimization objective.

In this paper, we present four contributions.

- **Speculative Expert Prefetching (SEP): HBM-to-cache prefetching for multi-GPU MoE serving.** To our knowledge, we are the first to address expert prefetching for production multi-GPU inference, where all experts reside in HBM and the only prefetch target of value is on-package cache (L2/LLC).

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

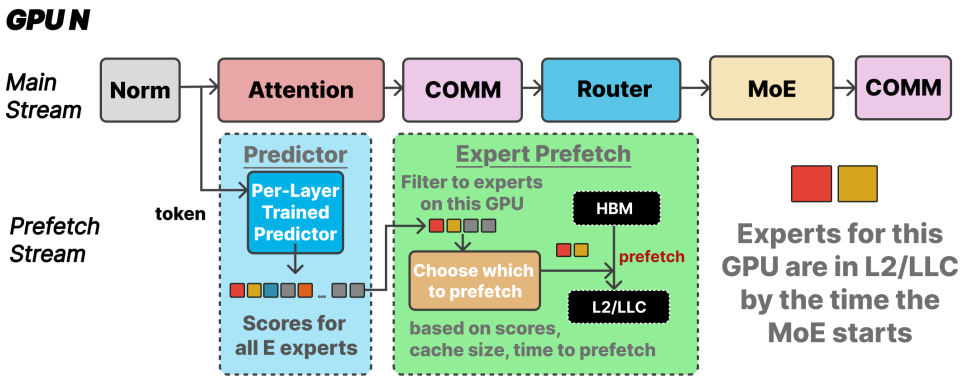


Figure 1: **Speculative Expert Prefetch:** A per-layer predictor reads the pre-attention hidden state and, concurrent with attention, communication, and routing, scores experts, maps to those this GPU owns under EP, and prefetches a cache-fit subset into L2/LLC before the expert GEMM executes.

- **A hardware-algorithm co-design principle for expert prefetching.** Cache capacity limits overprovisioning, reframing prediction from exact top-k matching to per-GPU coverage and ultimately setting the prefetcher's training objective. A second, temporal limit requires that the predictor and the expert weight transfer together complete before the expert GEMM begins. We measure both limits across four models to guide predictor and prefetch design.
- **A predictor design space characterization on production kernels.** We evaluate 18 predictor architectures across four MoE models for accuracy, memory, and latency, and measure decode-GEMM speedup on production MoE kernels against oracle and random-expert controls, reaching 1.3–1.8$\times$ speedup at batch 1 within the prefetch budget each model can deploy.
- **Train-once-per-model predictors that generalize to unseen datasets.** Predictors are trained once per model on a 42-dataset corpus spanning 13 task categories. On 22 held-out datasets from disjoint sources they reach 98.5–100% prediction accuracy, with no fine-tuning at deployment.

2 Background and Motivation

2.1 Memory-bound decode and the cache hierarchy.

An MoE layer routes each token through a router to its top-k of E experts; only those experts' weight matrices are read. Expert FFNs constitute ∼90–97% of model parameters (OpenAI, 2025; Jiang et al., 2024; DeepSeek-AI, 2024; 2025), and expert computation accounts for ∼30–70% of decode time (Park et al., 2026; Huang et al., 2025). Yet during decode, the expert GEMM performs negligible arithmetic relative to the weight traffic read from HBM, so bandwidth, not compute, sets its latency.

Modern GPU accelerators expose on-package memory tiers faster than HBM. AMD CDNA 3 and CDNA 4 accelerators place a 4 MB L2 cache on each of eight accelerator complex dies (XCDs), 32 MB in aggregate, backed by a 256 MB shared Infinity Cache (MALL) that acts as a last-level cache (Advanced Micro Devices, Inc., 2025a;b). NVIDIA Hopper and Blackwell accelerators provide GPU-wide L2 caches of 50 MB and 126 MB, respectively (NVIDIA Corporation, 2026a;c). Each tier is faster (L2 $>$ LLC $>$ HBM; L2 $>$ HBM on NVIDIA GPUs, which have no LLC) but smaller than the one below it, so staging expert weights in them before the GEMM issues directly reduces kernel latency. They are also far smaller than the expert weights that would occupy them (Table 1), which makes the prefetch target a matter of budget rather than placement.

2.2 The prefetch budget.

The prefetch budget is determined by two factors: **space** and **time**.

Table 1: Representative memory hierarchies in modern data-center GPUs (Advanced Micro Devices, Inc., 2025a;b;c; NVIDIA Corporation, 2026a;b;c). On-package caches offer higher bandwidth than HBM but orders of magnitude smaller capacity.

GPU	HBM	Infinity Cache (LLC)	L2
AMD MI325X OAM	256 GB	256 MB	32 MB
AMD MI355X OAM	288 GB	256 MB	32 MB
NVIDIA H100 SXM	80 GB	—	50 MB
NVIDIA B200 SXM	180 GB	—	126 MB

On the space constraint, the per-GPU prefetch budget N_{pf} is determined by cache capacity C and expert size S, so at most C/S experts fit. Table 2 reports this budget for the AMD Instinct MI355X, where the MALL holds between 3 and 42 experts per GPU depending on expert size, and the L2 between 0 and 5. Quantization scales this budget. An MXFP4 GPT-OSS-120B expert is roughly $4\times$ smaller than its bf16 counterpart, fitting proportionally more experts into the same capacity. The space budget is fixed by the model and hardware. Under expert parallelism, the GEMM additionally completes only when the *busiest* GPU has loaded its share, making per-GPU coverage at $(\text{EP}, N_{\text{pf}})$ the correct optimization target rather than global top-k accuracy, where EP is the number of GPUs and N_{pf} the number of experts prefetched per GPU.

Table 2: Per-GPU expert footprint and how many fit in each cache tier on AMD MI355X (8 GPUs)

Model	Format	Size (MB)	Exp/GPU	In L2	In MALL
GPT-OSS-120B	MXFP4	13	16	2	16
Mixtral-8×22B	bf16	76	8 (TP)	0	3
Qwen3-Next-80B	bf16	6	64	5	42
DeepSeek-V3.1	FP8	44	32	0	5

On the time constraint, the expert prefetch must be fully hidden under the operations that precede the MoE GEMMs. We write Δt for that overlap window, which begins at the attention block, whose layernorm output the predictor consumes alongside the QKV projection so the two start together, and ends where the expert GEMM begins, spanning the intervening operators, primarily the post-attention collective and routing, as pictured in Figure 2. Both the predictor and the transfer it enables must complete inside the window, so $T_{\text{predictor}} + T_{\text{transfer}} \leq \Delta t$. Δt therefore bounds how many experts can be prefetched, as opposed to just how many the cache can hold, and a deployment is held to whichever limit is smaller. Because the predictor must finish before expert identities are known, its latency is subtracted from the window before the transfer begins, so predictor capacity and prefetch coverage compete for the same window. We quantify Δt and its occupants for each model in Section 5.

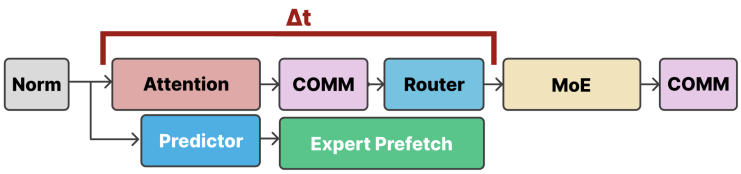


Figure 2: The overlap window Δt spans the attention block, the post-attention collective, and routing, ending at the expert GEMM. The predictor and transfer must both complete inside it.

3 SEP: Speculative Expert Prefetching

SEP addresses three questions jointly: **(1)** whether we can build a predictor that generalizes across MoE architectures and workloads; **(2)** which predictor architecture best fits the hardware latency and memory constraints; and **(3)** how the cache budget shapes the prediction objective and the prefetch mechanism. The subsections below address each in turn.

3.1 PRE-ATTENTION PREDICTION.

For each MoE layer ℓ we attach a small predictor g_ℓ that consumes the block's *pre-attention* hidden state x_ℓ and outputs a score over the layer's E experts. Because x_ℓ is ready before attention, g_ℓ and the resulting prefetch overlap attention, normalization, communication, and the router, all on the critical path. The predictor is purely speculative; the router still selects the experts that actually run, so a wrong prediction wastes prefetch bandwidth but never changes the model's output. The pre-attention formulation follows Zhu et al. (2025). We extend this to the HBM-to-cache prefetch scenario, characterizing the design choices required to deploy a predictor for GPU cache prefetching.

3.2 PREDICTOR TRAINING.

We build supervision by running each model decode-only in vLLM (Kwon et al., 2023) over our corpus and logging, per token and per layer, the pre-attention hidden state x_ℓ paired with the experts the router actually selects. We train *one predictor per layer*. Compared to Zhu et al. (2025), which trained and tested on a single dataset, our training draws from a 42-dataset corpus spanning 13 task categories; evaluation is conducted on 22 held-out datasets from the same categories but disjoint sources, as production serving systems encounter arbitrary input distributions. Details about the training and testing datasets are provided in Appendix E.

The OOD suite targets three levels of generalization: (1) nearby distribution shifts within the same task domain, such as harder math, science, and reading-comprehension benchmarks; (2) structural shifts, such as unseen programming languages, multilingual tasks, and new domains; and (3) stronger novelty tests using held-out or contamination-resistant benchmarks. Together, these evaluate whether performance transfers beyond the specific datasets and distributions seen during training, since each model's predictors are trained once and deployed unchanged.

3.3 PREDICTOR ARCHITECTURES.

We evaluate a broad family of predictor architectures to make predictor selection an empirical, reproducible design decision: a *linear* predictor; *MLPs* with one hidden layer at width $\{32, 128, 512, 2048\}$; *low-rank* ($r \in \{16, 128\}$) and *bottleneck* ($r \in \{32, 128\}$) factorizations; *mini-transformer* blocks ($d \in \{128, 256\}$); a *global* (cross-layer shared) variant; and the two architectures from Zhu et al. (2025), labeled by their distinguishing component, the ***BatchNorm-MLP*** (BatchNorm + GELU + dropout hidden layer) and the ***SiLU-MLP*** (Linear–SiLU–Linear). We extend these two architectures to also include a variable hidden size. The default MLP family uses a KL objective; the BatchNorm-MLP uses ranking-aware BCE (rBCE) and the SiLU-MLP uses per-expert weighted BCE (wBCE), following Zhu et al. (2025). More details about predictor architectures are presented in Appendix B.

3.4 PREDICTION METRICS.

Since N_{pf} exceeds the router's per-GPU top-k, spare cache slots are available to absorb prediction error. We exploit this by *overprovisioning*, prefetching the predictor's top-N_{pf} experts rather than its top-k, relaxing the prediction objective from exact set matching to coverage. Our headline metric is **per-GPU coverage at** $(\text{EP}, N_{\text{pf}})$: the fraction of a token's routed experts on the busiest GPU covered by the N_{pf} prefetched there, after EP/TP mapping. We write operating points as ep$E$_nN_{pf} (e.g., ep8_n2 = EP=8, N_{pf}=2/GPU), totaling $N_{\text{pf}} \times EP$ experts across all GPUs.

3.5 PREFETCH MECHANISM.

At inference time, the predictor evaluates x_ℓ on a side stream concurrent with attention, as a tuned HIP kernel. Predictor scores are mapped through the EP/TP placement to each GPU's expert subset; the top-N_{pf} are issued as asynchronous HBM-to-cache transfers of the full expert weight matrices, overlapping the post-attention collective and the router. Prefetching is a load-only pass of the same MoE kernel that later runs the GEMM. The mechanism is purely speculative; a cache miss falls back to the standard HBM read, leaving model outputs unaffected.

We prefetch the GEMM's own tiles (Figure 3). Each XCD's L2 is private and hardware places workgroups round-robin across dies, so a tile prefetched into the wrong XCD misses when the compute kernel runs. We therefore issue the top-N_{pf} experts through the same `xcd_swizzle` as the compute kernel, splitting each expert's N-tiles across the eight XCDs exactly as the GEMM will consume them. The MALL is shared, so it needs no such remap.

Because N_{pf} often exceeds what L2 holds, issue order matters. We prefetch in ascending score order, so the highest-scoring experts arrive last and are least likely to be evicted; those evicted from L2 remain in the MALL. In Figure 3, E2 and E15 stay in L2 while E102 is MALL-only.

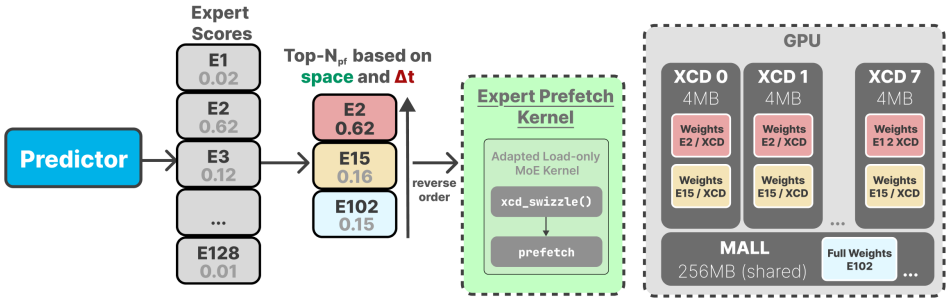


Figure 3: Selected expert weights prefetched into per-XCD L2 and shared MALL, issued in ascending score order through the compute kernel's `xcd_swizzle` and tiling. Residency is best effort.

The same `xcd_swizzle` might not be enough if the prefetch and compute kernels run on different streams. On a `torch.cuda.Stream` other than the compute stream, the round-robin origin can rotate, so `xcd_swizzle` might not map to the same physical XCD in both kernels. To solve this, we read the assigned XCD ID from an assembly instruction and permute the swizzle so both kernels will use the same XCD pattern.

Triton's `.ca` and `.cg` cache modifiers on `tl.load`/`tl.store` control cache retention (AMD, 2025b). `.ca` is the default, caching at all levels (L1, L2, and MALL) under LRU. `.cg` bypasses L1 and streams through L2; a line may hit in MALL but is neither inserted nor retained there, so weights are treated as streaming. On AMD CDNA, Triton lowers `.cg` to `sc0` plus the non-temporal bit. Prefetch therefore uses `.ca` so the later compute kernel can still hit those lines. Retention is best-effort, however. L2 is not software-controlled on MI355X, and activation traffic, a concurrent stream, or a later kernel can flush lines from both L2 and MALL.

4 Deployment Methodology and Scope

Models and layers. We evaluate four MoE models with distinct attention designs. GPT-OSS-120B alternates full and 128-wide sliding-window attention, Qwen3-Next-80B interleaves gated DeltaNet with full attention, Mixtral-8×22B uses grouped-query attention, and DeepSeek-V3.1 multi-head latent attention. Per-layer predictors are trained once per model (Section 3). We bound Δt to the narrowest layer types each model presents. For GPT-OSS-120B these are the sliding layers, whose fixed window makes their cost context-independent within the 2% variation we measure; for Qwen3-Next-80B, the gated-DeltaNet layers. Mixtral-8×22B and DeepSeek-V3.1 are uniform. Any predictor that fits the layers we time fits every layer of its model. We evaluate batch sizes 1 and 8 at a context length of 1024, the tightest case. Attention grows with context while the expert weight transfer does not, so longer contexts enlarge Δt and only add headroom. Across a sweep to 16k, no model's deployable N_{pf} changes.

Speedup measurements. All speedup measurements isolate the MoE operations for a single AMD Instinct MI355X GPU using production MoE kernels from AMD's open-source ROCm AITER library (AMD, 2025a). The kernel is benchmarked in two configurations, cache cold and prefetched cache, using HIP events and `rocprof` traces. Attention and dense layers are tensor-parallel across the node's eight GPUs. Experts are expert parallel, so each GPU owns whole experts, except Mixtral-8×22B, whose experts are tensor parallel and leave every rank holding a slice of all eight.

A random-expert control prefetches experts without a predictor, to isolate performance gains from prefetching by chance versus using a predictor. An oracle predictor is also introduced, to compare our results against a predictor which has 100% accuracy.

Inference kernel measurements. We serve each model in vLLM on a single MI355X node and time the kernels it dispatches, cross-checking the configuration against vLLM's published recipes and AMD's ROCm tuning guide. Attention resolves to a unified attention kernel, the tensor-parallel collective to a custom IPC all-reduce, and the linear layers to tuned GEMMs, all from AITER. Inference kernel timings are taken under HIP graph capture with HIP events, and predictor and prefetch times with an in-kernel GPU clock.

5 Results and Discussion

5.1 Does prediction generalize, and what cache budget does each model get?

OOD accuracy is uniformly high; what varies across models is the cache tier their expert size forces. Table 3 reports OOD prediction accuracy at each model's cache-fit operating point. A compact per-layer predictor achieves 98.5–100% accuracy across all four models; even a linear model falls within three percentage points on three of them. Outside Section 5.3 we report the linear predictor and the SiLU-MLP, the smallest and the most accurate of the 18 architectures evaluated there. Expert size, determined by expert count per GPU and quantization format, governs which cache tier holds a useful number of experts and hence sets the prefetch budget N_{pf}. Qwen3-Next (6 MB, bf16) and GPT-OSS (13 MB, MXFP4) are small enough for L2 to hold a useful number of experts, so we operate at $N_{\mathrm{pf}}{=}4$ and 2 (Table 2). DeepSeek (44 MB, FP8) does not fit a full expert into L2. Its budget shifts to the MALL, where 5 experts fit and prediction accuracy rises from 63.5% to 100%. Mixtral (76 MB per GPU, bf16) places 3 resident experts in MALL.

Table 3: OOD prediction accuracy at each model's cache-fit operating point (EP=8; Mixtral-8×22B served TP=8). Expert size determines the cache tier and prefetch budget N_{pf}. Results are shown for the linear predictor and the best SiLU-MLP. DeepSeek-V3.1 is reported at both tiers to illustrate the necessary shift from L2 to MALL when expert size exceeds L2.

Model	Tier	Pref./GPU	Linear	**SiLU-MLP (best)**
GPT-OSS-120B	L2	2	96.9%	**98.5%**
Mixtral-8×22B	MALL (TP)	3 of 8	95.3%	**98.7%**
Qwen3-Next-80B	L2	4	97.3%	**98.8%**
DeepSeek-V3.1	L2	1	56.4%	63.5%
DeepSeek-V3.1	MALL	5	97.6%	**100.0%**

5.2 What fits inside the overlap window?

Every model evaluated except Mixtral fits the largest predictor, but only DeepSeek can prefetch its full cache capacity. Both the predictor and the transfer must complete inside the window of Section 2.2:

$$T_{\text{predictor}} + T_{\text{transfer}} \;\leq\; \Delta t. \tag{1}$$

We capture latencies across models on 8 GPUs at batch size 1 decode to identify which predictor and prefetch budget fit the window (Section 4). Table 4 normalizes them to each model's own Δt, where below 100% indicates the predictor and prefetch operations fit within that model's window.

Three of four models fit comfortably at their operating point. GPT-OSS, Qwen3-Next, and DeepSeek absorb the SiLU-MLP predictor together with a full cache-fit prefetch in 40–46% of Δt. At these budgets timing does not constrain predictor choice, which is governed instead by the cache-fit budget of Table 2 and the accuracy–latency trade-off of Section 5.3. Timing does constrain larger budgets, however. GPT-OSS and Qwen3-Next exceed the window before reaching their full MALL capacity, and Figure 5 marks each model's limit. DeepSeek does not, because its window is roughly $3.4\times$ theirs.

Mixtral alone cannot prefetch its cache-fit budget in time. Its experts are 76 MB per GPU, $12\times$ Qwen3-Next's. At batch 1 the transfer alone exceeds the window at the $N_{\mathrm{pf}}{=}3$ its cache holds; at

N_{pf}=2, its top-k, the linear predictor fits at 92% while SiLU-MLP overruns at 113%. It is thus the one model where the predictor is chosen for latency rather than accuracy. For batch 8, Δt grows 41%, since the projections scale with token count while the latency-bound stages do not, and each predictor gains one expert of budget: linear reaches N_{pf}=3, SiLU-MLP reaches 2. Predictor latency rises with batch, but so does the window, and what matters is the room left for the transfer. That room grows 42% for Mixtral, which is why it gains budget as we increase the batch size, while for GPT-OSS it falls 25%, its window staying nearly flat while the SiLU-MLP nearly doubles.

Table 4: Predictor and transfer latencies normalized to each model's own overlap window Δt, measured on MI355X across 8 GPUs at batch size 1 decode. Δt spans the attention block through the post-attention collective and routing. SEP operations below 100% fit. The transfer is taken at the N_{pf} each model's cache holds (Table 2).

	GPT-OSS	Qwen3-Next	Mixtral	DeepSeek
SEP operations (% of Δt)				
Linear predictor	13	10	17	6
SiLU-MLP predictor	26	24	38	13
Transfer at cache-fit N_{pf}	20	17	**115**	33
(N_{pf})	(2)	(4)	(3)	(5)
Linear + transfer	33	27	**132**	39
SiLU-MLP + transfer	46	40	**153**	46

5.3 Which predictor should you use?

A linear predictor where the prefetch budget substantially exceeds the router's top-k, the SiLU-MLP where it does not, and on Mixtral only what the window allows. Figure 4 characterizes the accuracy-vs-cost trade-off for GPT-OSS across all 18 predictor architectures at the cache-fit operating point (ep8_n2), where accuracy measures the fraction of the router's top-k experts covered within the overprovisioned prefetch budget N_{pf}. The accuracy frontier rises steeply then saturates. A linear model achieves 96.9% at 0.37 M parameters, while the best predictor, the SiLU-MLP at h=2048, reaches 98.5% at 6.2 M. Appendix C reports the same sweep for Qwen3-Next and DeepSeek (Table 5).

The SiLU-MLP leads at every size and returns saturate by h=1024. The cross-layer-shared global variant matches the per-layer MLP at the same width, indicating that layer-sharing incurs negligible accuracy cost. Across all four models (Table 3), the linear-to-best gap ranges from 1.5 to 3.4 points, widest on Mixtral, which has fewer experts per GPU to choose from. The choice of predictor interacts with the overprovisioning budget. When the cache-fit budget N_{pf} exceeds the router's top-k by a wide margin, spare cache slots absorb prediction errors, making a cheaper, less accurate predictor viable. A linear predictor may therefore suffice in settings with generous overprovisioning, while tighter budgets justify using a larger predictor such as the SiLU-MLP. On GPT-OSS, Qwen3-Next, and DeepSeek the timing budget does not narrow this choice further. The SiLU-MLP at h=2048, the most accurate architecture, occupies only 13–26% of Δt (Table 4), so the accuracies in Table 3 are attainable at each model's operating point.

In the case of Mixtral, prefetching its 76 MB experts limits the time budget left for the predictor (Section 5.2). Where the prefetch is large relative to the window, predictor capacity must be traded for prefetch coverage. In Section 5.4 we show the performance implications of this trade-off.

5.4 What is the performance improvement when using expert prefetchers?

1.3–1.8× *decode-GEMM speedup at batch 1, within the prefetch budget each model can deploy.* Figure 5 reports decode-GEMM speedup versus prefetch budget N_{pf} at batch sizes 1 and 8, on production AITER MoE kernels, relative to a MoE GEMM without prefetching. Each panel presents four curves: the predicted (Linear, SiLU-MLP) prefetches, a perfect-prediction oracle, and a random-expert control that isolates the benefit of a predictor-driven prefetch decision. Every panel is overlaid with its time limit, where a dashed line marks the largest N_{pf} each predictor can deploy and shading marks budgets neither can reach (Section 5.2). Reported speedups are for the best predictor each model can deploy.

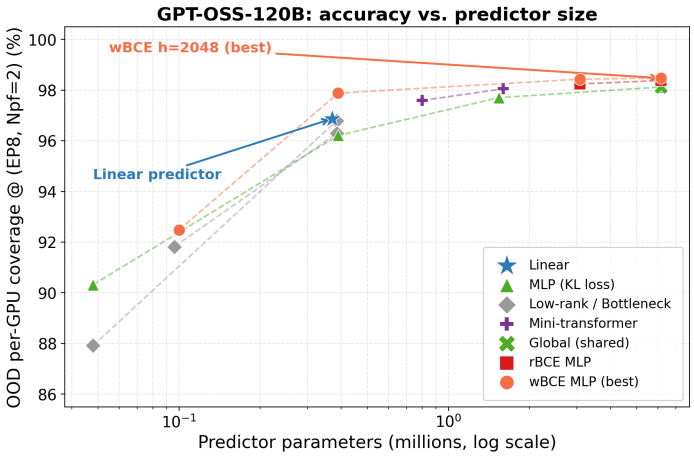


Figure 4: Prediction accuracy on held-out workloads vs. predictor size for GPT-OSS (18 predictors) at the cache-fit operating point ep8_n2 and batch size 1, with x the parameter count on a log scale. The linear predictor achieves 96.9% at 0.37 M, while the SiLU-MLP leads at 98.5% (6.2 M).

The predicted prefetch tracks the oracle closely and attains near-ceiling speedup within the cache-fit budget. For GPT-OSS, $N_{\text{pf}}{=}2$ reaches $\sim 1.46\times$ at batch 1, while the random control needs $8\times$ more prefetches ($N_{\text{pf}}{=}16$, $1.40\times$) to approach it; the gap between the two is the value of prediction, not of prefetching. Qwen3-Next reaches $\sim 1.45\times$ at $N_{\text{pf}}{=}4$ (batch 1) and $\sim 1.63\times$ at $N_{\text{pf}}{=}32$ (batch 8), and DeepSeek $\sim 1.30\times$ at $N_{\text{pf}}{=}5$. Mixtral reaches $\sim 1.78\times$ at batch 1, but only with the linear predictor at $N_{\text{pf}}{=}2$, as the SiLU-MLP does not fit that budget (Section 5.2), where it would have reached $1.91\times$ against a $2.01\times$ oracle.

Larger batches require a larger N_{pf} to sustain per-GPU coverage. At batch 8 GPT-OSS requires $N_{\text{pf}}{=}8$ rather than 2 to reach $\sim 1.71\times$, since more tokens activate more distinct experts against a fixed cache. At that operating point GPT-OSS separates a $1.17\times$ random control from a $1.73\times$ oracle, whereas Mixtral's eight experts and top-2 routing mean batch 8 activates nearly all of them, collapsing the random control ($1.37\times$) onto the oracle ($1.41\times$) at $N_{\text{pf}}{=}3$. Prediction quality matters where the expert pool is large relative to what fits in cache. Batching also widens Δt (Section 5.2), and that extra room is better spent on prefetch budget than on predictor capacity. At batch 8 Mixtral can afford either $N_{\text{pf}}{=}3$ with the linear predictor or $N_{\text{pf}}{=}2$ with the SiLU-MLP, and the former is worth $1.40\times$ against the latter's $1.26\times$.

6 Related Work

6.1 Cache and Memory Prefetching.

The most directly related hardware work is PRESERVE (Yüzügüler et al., 2025), which overlaps HBM-to-cache weight staging with inter-device communication. PRESERVE targets deterministically-known tensors and requires no predictor; the MoE expert GEMM is data-dependent, necessitating the speculative prediction mechanism we introduce.

6.2 MoE Expert Prediction and Prefetch.

The dominant approach in prior work stages experts from CPU DRAM or SSD into GPU DRAM, either reactively (Eliseev & Mazur, 2023; Hwang et al., 2024; Yi et al., 2023; Zhong et al., 2024; Yu et al., 2024; Tang et al., 2024; Kong et al., 2024) or via learned prediction (Xue et al., 2024; Song et al., 2024; He et al., 2024; Yu et al., 2025; Yan et al., 2025; Fang et al., 2025b); recent speculative-prefetch systems target the same host↔device offload scenario (Chen et al., 2025; Zhang et al., 2025b; Wang et al., 2025). SEP instead focuses on deployments where all experts reside in HBM and prefetching targets on-package L2/LLC. FATE (Fang et al., 2025a) predicts a later layer's routing from an earlier layer's gate input; we reimplement FATE's cross-layer formulation, using

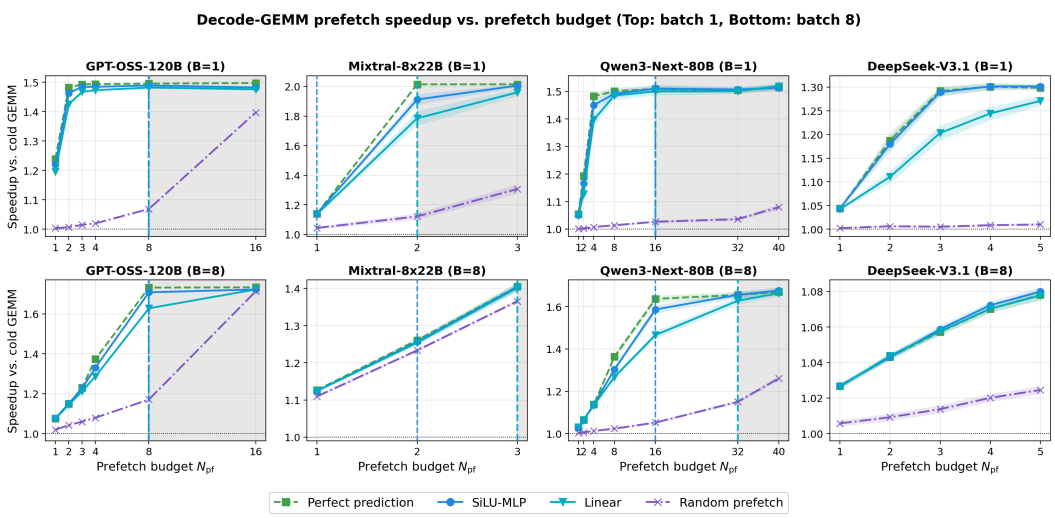


Figure 5: Decode-GEMM speedup versus prefetch budget N_{pf} at batch sizes 1 (top) and 8 (bottom). Each panel shows both predictors, a perfect-prediction oracle, and a random-expert control, with line color matching the predictor. DeepSeek is unshaded as its window admits every budget plotted.

the pre-attention input from layer $N-1$ to predict experts in layer N with the same architectures. Our smaller linear predictor achieves higher recall@4 accuracy on GPT-OSS-120B (82.5%) than the larger cross-layer MLP (78.7%). The pre-attention predictor of (Zhu et al., 2025) evaluates two predictor architectures on a single dataset; we extend this to a systematic study of 18 architectures across four modern MoE models and 22 held-out evaluation datasets. In (Pei et al., 2025), router logits are used to retain frequently-activated experts in memory; this frequency-based retention is reactive and cannot overlap the expert load with the attention pipeline. Popularity-based static caching (Zhang et al., 2025a) achieves only $\approx$35.5% prediction accuracy when we prefetch the top-2 most popular experts per layer on GPT-OSS-120B versus 98.5% for our SiLU-MLP predictor, confirming that expert routing is highly input-dependent.

7 Conclusion

SEP addresses the memory-bandwidth bottleneck of MoE decode by predicting each token's active experts from the pre-attention hidden state and staging their weights into on-package cache before the expert GEMM executes. We identify that the correct optimization objective in the all-in-HBM multi-GPU deployment is per-GPU coverage within the cache budget, not global top-k accuracy. A design space exploration across 18 predictor architectures and four modern MoE models demonstrates that a compact per-layer predictor achieves 98.5–100% prediction accuracy, with a linear predictor within three percentage points on three of the models. These predictors generalize to unseen datasets, establishing production viability. The predictor architecture and prefetch budget together constitute a tunable design knob that adapts SEP to the memory hierarchy and latency constraints of any target deployment. The space and time limits can pull in opposite directions. When the cache-fit budget affords sufficient overprovisioning, spare slots absorb mispredictions and a linear predictor suffices, while tighter overprovisioning warrants a larger, more accurate predictor such as the SiLU-MLP. The overlap window imposes the reverse pressure. Where expert weights are large relative to that window, the transfer consumes it and predictor capacity must be given up to preserve prefetch coverage. Predictor choice and prefetch budget are therefore co-optimized, conditioned on the model's kernels and the accelerator they run on. Both limits move as expert size, quantization format, and cache hierarchy change. On MI355Xs with production MoE kernels from AMD's open-source ROCm AITER library at batch 1, SEP yields 1.3–$1.8\times$ decode-GEMM speedups.

REPRODUCIBILITY STATEMENT

We provide implementation details, including the experimental setup and hardware configurations, in Sections 3 and 4. All datasets used in our experiments are listed in Table 6. We additionally describe the evaluation and benchmarking procedures needed to reproduce the main results.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code for dataset extraction and preprocessing, training loops, initial drafts of prefetch kernels, and benchmarking scripts for performance and accuracy. Generative AI tools were also used to assist with drafting portions of the manuscript and brainstorming dataset selection to improve experimental coverage.

We did not use generative AI tools to develop the core MoE kernels, formulate the main research hypotheses or conceptual framework, or interpret the reported performance results and GPU traces. Other required-disclosure tasks, including mathematical claims and proofs, translation, and qualitative analysis, were not applicable to this work.

All AI-assisted work was reviewed and verified by the authors. AI-assisted dataset extraction was checked against the original datasets, and AI-generated or modified code for dataset extraction, prefetch kernels, and benchmarking was inspected and tested for correctness. Experimental results and performance conclusions were independently validated by the authors, including through examination of the corresponding GPU traces where applicable.

We take responsibility for the final content of this work, including all text, claims, results, code, and artifacts produced with the aid of generative AI.

REFERENCES

Advanced Micro Devices, Inc. AMD CDNA 3 Architecture: The All-New AMD GPU Architecture for the Modern Era of HPC and AI. Architecture white paper, Advanced Micro Devices, Inc., 2025a. URL https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf. Accessed: 2026-07-25.

Advanced Micro Devices, Inc. Introducing AMD CDNA 4 Architecture. Architecture white paper, Advanced Micro Devices, Inc., 2025b. URL https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-4-architecture-whitepaper.pdf. Accessed: 2026-07-25.

Advanced Micro Devices, Inc. AMD Instinct MI325X Accelerator. Data sheet, Advanced Micro Devices, Inc., 2025c. URL https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/product-briefs/instinct-mi325x-datasheet.pdf. Accessed: 2026-09-03.

AMD. AITER: AI tensor engine for ROCm. https://github.com/ROCm/aiter, 2025a. GitHub repository.

AMD. Iris: Triton device functions reference. https://rocm.github.io/iris/reference/triton/device-functions.html, 2025b. ROCm documentation. Accessed: 2026-09-08.

Liangkun Chen, Zijian Wen, Tian Wu, Xiaoxi Zhang, and Chuan Wu. SP-MoE: Speculative decoding and prefetching for accelerating MoE-based model inference, 2025.

DeepSeek-AI. DeepSeek-V3 technical report, 2024.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Artyom Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading, 2023.

Zhiyuan Fang, Zicong Hong, Yuegui Huang, Yufeng Lyu, Wuhui Chen, Yue Yu, Fan Yu, and Zibin Zheng. Fate: Fast edge inference of mixture-of-experts models via cross-layer gate, 2025a.

Zhiyuan Fang, Yuegui Huang, Zicong Hong, Yufeng Lyu, Wuhui Chen, Yue Yu, Fan Yu, and Zibin Zheng. Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline, 2025b.

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), 2022.

Xin He, Shunkang Zhang, Kaijie Tang, Shaohuai Shi, Yuxin Wang, Zihao Zeng, Zhenheng Tang, Xiaowen Chu, Haiyan Yin, Ivor W. Tsang, and Yew Soon Ong. ExpertFlow: Efficient mixture-of-experts inference via predictive expert caching and token scheduling, 2024.

Zongle Huang, Lei Zhu, Zongyuan Zhan, Ting Hu, Weikai Mao, Xianzhi Yu, Yongpan Liu, and Tianyu Zhang. Moesd: Unveil speculative decoding's potential for accelerating sparse moe. *arXiv preprint arXiv:2505.19645*, 2025.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. Pre-gated MoE: An algorithm-system co-design for fast and scalable mixture-of-expert inference. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

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, 2024.

Rui Kong et al. SwapMoE: Serving off-the-shelf MoE-based large language models with tunable memory budget, 2024.

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

NVIDIA Corporation. NVIDIA Blackwell Tuning Guide. `https://docs.nvidia.com/cuda/archive/12.8.0/pdf/Blackwell_Tuning_Guide.pdf`, 2026a. CUDA Toolkit 12.8 documentation, accessed: 2026-09-23.

NVIDIA Corporation. NVIDIA HGX AI Factory (HGX H100, H200, B200): Components. `https://docs.nvidia.com/enterprise-reference-architectures/hgx-ai-factory-h100-h200-b200/latest/components.html`, 2026b. Enterprise reference architecture, accessed: 2026-09-23.

NVIDIA Corporation. NVIDIA Hopper Tuning Guide. `https://docs.nvidia.com/cuda/hopper-tuning-guide/index.html#increased-l2-capacity`, 2026c. CUDA Toolkit documentation, accessed: 2026-07-25.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025.

Jongseok Park, Sunga Kim, Zhenyu Gu, Ion Stoica, and Alvin Cheung. Uncovering intra-expert activation sparsity for efficient mixture-of-expert model execution. *arXiv preprint arXiv:2605.08575*, 2026.

Zehua Pei, Ying Zhang, Hui-Ling Zhen, Tao Yuan, Xianzhi Yu, Zhenhua Dong, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. PreMoE: Proactive inference for efficient mixture-of-experts, 2025.

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. In *International Conference on Learning Representations (ICLR)*, 2017.

Xiaoniu Song, Zihang Zhong, Rong Chen, and Haibo Chen. ProMoE: Fast MoE-based LLM serving using proactive caching, 2024.

Peng Tang et al. HOBBIT: A mixed precision expert offloading system for fast MoE inference, 2024.

Wenfeng Wang, Jiacheng Liu, Xiaofeng Hou, Xinfeng Xia, Peng Tang, Mingxuan Zhang, Chao Li, and Minyi Guo. MoE-SpeQ: Speculative quantized decoding with proactive expert prefetching and offloading for mixture-of-experts, 2025.

Leyang Xue, Yao Fu, Zhan Lu, Chuanhao Sun, Luo Mai, and Mahesh Marina. MoE-Infinity: Efficient MoE inference on personal machines with sparsity-aware expert cache, 2024.

Jiaming Yan, Jianchun Liu, Hongli Xu, and Liusheng Huang. Accelerating mixture-of-expert inference with adaptive expert split mechanism, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report, 2025.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. EdgeMoE: Fast on-device inference of MoE-based large language models, 2023.

Dianhai Yu, Liang Shen, Hongxiang Hao, Weibao Gong, Huachao Wu, Jiang Bian, Lirong Dai, and Haoyi Xiong. MoESys: A distributed and efficient mixture-of-experts training and inference system for internet services, 2024. URL `https://arxiv.org/abs/2205.10034`.

Hanfei Yu, Xingqi Cui, Hong Zhang, Hao Wang, and Hao Wang. fMoE: Fine-grained expert offloading for large mixture-of-experts serving, 2025.

Ahmet Caner Yüzügüler, Jiawei Zhuang, and Lukas Cavigelli. PRESERVE: Prefetching model weights and KV-cache in distributed LLM serving, 2025.

Junyi Zhang et al. PopFetcher: Towards accelerated mixture-of-experts training via popularity based expert-wise prefetch. In *2025 USENIX Annual Technical Conference (USENIX ATC)*, 2025a.

Yuning Zhang, Grant Pinkert, Nan Yang, Yanli Li, and Dong Yuan. DuoServe-MoE: Dual-phase expert prefetch and caching for LLM inference QoS assurance, 2025b.

Shuzhang Zhong, Ling Liang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. AdapMoE: Adaptive sensitivity-based expert gating and management for efficient MoE inference. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.

Shien Zhu, Samuel Bohl, Robin Oester, and Gustavo Alonso. Pre-attention expert prediction and prefetching for mixture-of-experts large language models, 2025.

A BUSIEST-GPU DEMAND

Under expert parallelism a layer's experts are sharded across GPUs, and the expert GEMM reduction is a synchronization point; it completes only when the *busiest* GPU has finished its share. Until that point, the reduction step cannot be completed.

Figure 6 gives this distribution for GPT-OSS-120B (router top-4) at three expert-parallel degrees. The demand is small and EP-dependent: at EP=8 (16 experts per GPU) the busiest GPU holds $\leq$2 of a token's experts about 97.5% of the time, so a budget of N_{pf}=2 can cover almost every token if the right experts are staged. Reducing EP packs more experts onto each GPU and shifts the distribution upward. At EP=2 (64 experts per GPU) the mode moves to 2–3 experts per GPU—so lower-EP deployments need a larger N_{pf} to reach the same coverage. This is why we report per-GPU coverage at an explicit $(\mathrm{EP}, N_{\mathrm{pf}})$ operating point (Section 2) rather than a single global accuracy number.

B PREDICTOR ARCHITECTURES

All predictors are per-layer and consume the pre-attention hidden state $x_\ell \in \mathbb{R}^H$, producing a score over the layer's E experts. We sweep 18 architectures. Below, H is the model hidden size and E the number of experts; parameter counts therefore differ by model.

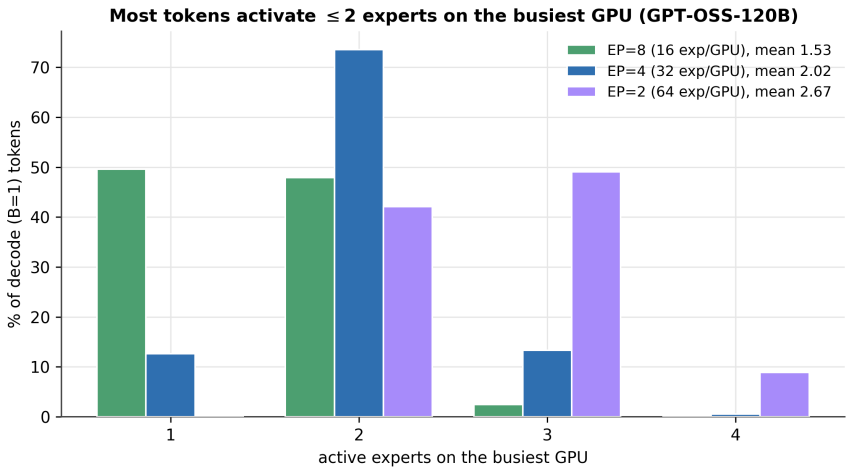


Figure 6: **Most tokens need only a few experts on the busiest GPU.** Empirical distribution of a token's routed experts landing on the *busiest* GPU per decode step (GPT-OSS-120B, router top-4, using 1000 random prompts) at EP=8/4/2. Demand grows with router top-k and shrinks with EP: at EP=8 most tokens need $\leq$2 experts per GPU, while lower EP shifts demand upward.

- **Linear**: a single $H \times E$ projection. Params $= HE + E$. The simplest predictor and a strong baseline.
- **MLP (KL loss)** at hidden width $h \in \{32, 128, 512, 2048\}$: one hidden layer with GELU, trained with a KL objective against the router distribution. Params $= Hh + h + hE + E$.
- **Low-rank** ($r \in \{16, 128\}$): factorized projection $H \to r \to E$ with no nonlinearity. Params $= Hr + rE + E$.
- **Bottleneck** ($r \in \{32, 128\}$): an MLP whose hidden width is the bottleneck r (same form as the MLP family). Params $= Hr + r + rE + E$.
- **Mini-transformer** ($d \in \{128, 256\}$): a small attention block over a short pre-attention context, then a projection to E.
- **Global** (h=2048): a single shared predictor across *all* layers (layer index as a feature) rather than one predictor per layer.
- **BatchNorm-MLP** and **SiLU-MLP** ($h \in \{32, 128, 1024, 2048\}$): the same two-layer MLP backbone trained with a pairwise ranking BCE and a per-expert weighted BCE loss respectively. These two losses were introduced for pre-attention expert prediction by Zhu et al. (2025); we credit that work and rename the variants by their architecture. Params match the MLP family at the same h.

For reference, on GPT-OSS-120B (H=2880, E=128): Linear $=$ 0.37M; MLP/BatchNorm-MLP/SiLU-MLP at h=32/128/512/1024/2048 $=$ 0.10/0.39/1.54/3.08/6.16M. Larger-H models (Mixtral-8$\times$22B H=6144, DeepSeek-V3.1 H=7168) have proportionally larger predictors; Qwen3-Next-80B (H=2048, E=512) is dominated by the output projection.

C Full out-of-distribution accuracy by architecture

Table 5 gives OOD accuracy for all 18 architectures on three models, each at its cache-fit operating point (GPT-OSS-120B ep8_n2; Qwen3-Next-80B/DeepSeek-V3.1 ep8_n4 per-GPU coverage). The SiLU-MLP at h=2048 is best on every model; the linear predictor remains competitive despite its simplicity.

Table 5: OOD accuracy (%) by architecture and model, at the cache-fit operating point. GPT-OSS-120B: ep8_n2; Qwen3-Next-80B/DeepSeek-V3.1: ep8_n4 (per-GPU coverage). Multi-seed (42/43/44) spread is < 0.2 pts.

Architecture	GPT-OSS-120B	Qwen3-Next-80B	DeepSeek-V3.1
Linear	96.87	96.02	94.29
MLP-KL (h=32)	90.32	79.52	67.32
MLP-KL (h=128)	96.21	91.39	86.46
MLP-KL (h=512)	97.70	95.87	96.49
MLP-KL (h=2048)	98.12	97.36	98.51
Low-rank (r=16)	87.92	73.72	59.77
Low-rank (r=128)	96.79	93.53	88.05
Bottleneck (r=32)	91.82	82.35	69.43
Bottleneck (r=128)	96.29	91.19	86.40
Mini-transformer (d=128)	97.59	93.73	89.57
Mini-transformer (d=256)	98.05	96.07	97.30
Global (h=2048)	98.12	97.36	98.47
BatchNorm-MLP (h=1024)	98.24	97.33	99.05
BatchNorm-MLP (h=2048)	98.38	98.03	99.41
SiLU-MLP (h=32)	92.48	81.02	68.63
SiLU-MLP (h=128)	97.88	93.18	90.37
SiLU-MLP (h=1024)	98.43	98.15	99.61
SiLU-MLP (h=2048)	98.47	98.37	99.66

D PER-LAYER COVERAGE

Figure 7 shows per-GPU coverage as a function of MoE layer index for the SiLU-MLP and the linear predictor. Coverage is high and relatively stable across the network. The SiLU-MLP outperforms the linear predictor at every layer, most visibly on the larger-expert models, though both remain competitive across the layers.

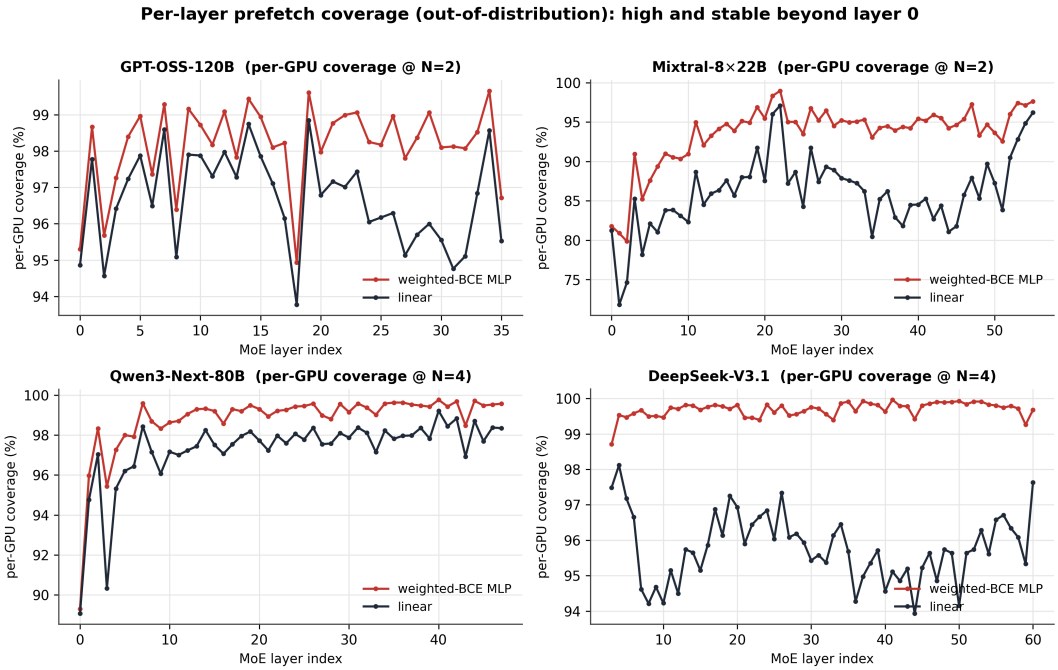


Figure 7: Per-layer per-GPU coverage (OOD) at each model's headline N_{pf}. The SiLU-MLP (red) is uniformly above the linear predictor (dark).

E DATASETS AND LICENSING

Routing captures are drawn from a 42-dataset corpus spanning 13 categories: general/open-domain QA, reading comprehension, code, mathematics, commonsense and multi-step reasoning, multilingual text, dialogue/chat, summarization, retrieval/RAG, science, instruction-following/safety, long-context, and structured/tabular inputs. Training uses a subset; evaluation is on *22 held-out datasets* drawn from the same categories but disjoint from training, as well as domain-shift categories. All sources are standard public benchmarks used under their respective licenses (predominantly MIT, Apache-2.0, and CC-BY variants); no private or human-subjects data are used.

Table 6: Full dataset manifest. Training corpus (42 datasets, 13 categories; code is one category shown as two rows) and the held-out out-of-distribution suite (22 datasets, 8 categories, disjoint from training). All are public benchmarks used under their respective licenses.

Category	Datasets
Training corpus — 42 datasets, 13 categories	
Math	gsm8k, math, mgsm, aqua_rat, asdiv
Code (Python)	humaneval, mbpp, ds1000, bigcodebench
Code (other langs.)	multipl_e_{java, js, go, cpp}
Commonsense / MC	arc_easy, openbookqa, winogrande, hellaswag
Reasoning	bbh, agieval
Reading comprehension	squad_v2, boolq, drop
Open-domain QA	trivia_qa, natural_questions
Chat / instruction	dolly, oasst1, wildbench, mt_bench, sharegpt, hh_rlhf_helpful
Summarization	cnn_dailymail, xsum, samsum, gov_report, narrativeqa_passages, billsum
Dialog	blended_skill_talk
Translation	opus100_translation
Structured (SQL)	spider
Creative	writing_prompts
Raw web / encyclopedic	c4_en_sample, wikipedia_en_paragraphs
Out-of-distribution evaluation — 22 datasets, 8 categories	
Math (held out)	gsm_hard
Code (held-out langs.)	multipl_e_{rust, typescript, ruby}
Code (contam.-resistant)	livecodebench, cruxeval
Science / MC	arc_challenge, mmlu, mmlu_pro, scienceqa
Reading comp. / NLI	race, multinli, freebaseqa
Open-domain reasoning	hotpotqa
Chat / preference	oasst2, mt_bench_2
Domain shift	pubmedqa, fiqa, truthfulqa, xnli_fr, xcopa_et, govreport_long