ActiveEvict: Budget-Aware Pre-Eviction for Efficient MoE Offloading

Anonymous ACL submission

Abstract

Mixture-of-Experts (MoE) architectures enable scaling Large Language Models (LLMs) by decoupling model capacity from computation. However, their large parameter footprint makes expert offloading to host memory necessary, creating I/O-bound inference bottlenecks. Existing methods rely on prefetching to hide latency but remain limited by short computation windows and passive one-for-one eviction under static budgets. We observe that coupling eviction and loading on the critical path causes frequent pipeline stalls and poor cache utilization. To address this, we propose ACTIVEEVICT, a framework that proactively evicts experts and performs budget-aware routing, transforming static memory budgets into dynamic effective budgets. This decoupling reduces I/O stalls and enables better GPU memory utilization. Experiments show that ACTIVEEVICT reduces blocking I/O time by up to 46% compared to state-of-the-art prefetch methods, while incurring less than 1% accuracy loss, demonstrating significant throughput improvement under constrained memory.

1 Introduction

Large Language Models (LLMs) have achieved remarkable advances in natural language understanding and generation (Touvron et al., 2023; Chiang et al., 2023; Chowdhery et al., 2023; Zhang et al., 2022), largely driven by their massive parameter scales. However, the growing model size significantly increases the computational and memory requirements for inference, posing a major challenge for efficient deployment on GPUs or edge devices.

Sparse activation architectures, such as Mixture-of-Experts (MoE) (Shazeer et al., 2017), address this challenge by incorporating a large set of parallel expert networks while activating only a small subset per token. This design decouples model capacity from computation, enabling LLMs to scale effectively while maintaining high performance on tasks including question answering, text generation, and machine translation (Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Pope et al., 2023).

Despite the computational efficiency of MoE-based LLMs, their full parameter sets often exceed the memory capacity of standard GPUs. For example, Mixtral-8×7B (Jiang et al., 2024a) requires roughly 87GB to store all parameters, while only 14 billion are active per token. In memory-constrained scenarios, expert offloading to host memory has become standard (Eliseev and Mazur, 2023; Wei et al., 2024). However, the latency of moving experts over PCIe or other interconnects often dominates inference, creating severe I/O-bound bottlenecks that limit overall throughput.

Prefetching-based strategies (Hwang et al., 2024; Tang et al., 2026; Kamahori et al., 2024; Xue et al., 2024) attempt to mitigate I/O stalls by loading experts before they are needed, overlapping communication with computation. In practice, these approaches are limited by the short computation windows available in non-expert layers and by the reliance on accurate expert prediction. Mispredictions lead to redundant evictions and loads, which exacerbate pipeline stalls and further degrade performance.

A fundamental limitation of existing methods is their reliance on static memory budgets for expert caching, combined with passive, one-for-one eviction policies. With a static cache budget, eviction and loading are tightly coupled on the inference critical path, causing each cache miss to incur both swapping-out and swapping-in latency, and preventing timely reclamation of GPU memory for low-utility experts.

To address these limitations, we propose a novel MoE inference framework that combines Proactive Expert Eviction with Budget-Aware Routing. As illustrated in Figure 1, our design decouples expert eviction from the loading path by moving the

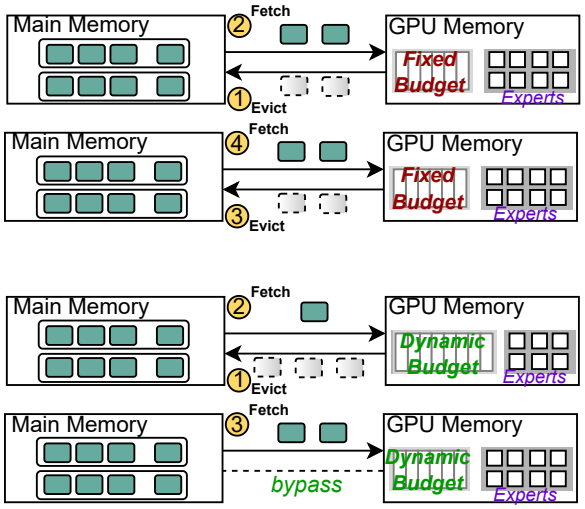


Figure 1: Overview of our pre-eviction and dynamic-budget design for MoE offloading inference. Pre-eviction allows GPU memory to be reclaimed, turning the expert cache budget from fixed to dynamic and enabling budget-aware expert activation.

eviction process ahead of the routing stage. This transforms the static cache budget into a dynamic effective budget that fluctuates based on the execution state. This proactive mechanism allows the system to flexibly regulate the expert cache: reclaiming memory early during high-pressure periods while retaining on-card experts whenever the load allows. By leveraging this proactive eviction, the I/O stalls following routing decisions are significantly reduced. Building on the dynamic budget, we further adapt expert activation decisions during routing to align with available GPU memory, effectively alleviating the I/O bottleneck caused by expert swapping while preserving inference quality.

Our main contributions are as follows:

- We identify the limitations of fixed memory budgets in MoE offloading and demonstrate how passive eviction policies lead to critical-path I/O stalls.
- We propose a dynamic budget management framework that enables proactive memory reclamation and efficient expert swapping.
- Experiments on Mixtral-8×7B and LLaMA-MoE show ACTIVEEVICT reduces blocking I/O time up to 46% compared to state-of-the-art prefetch methods, while incurring less than 1% accuracy loss.

2 Background & Motivation

2.1 Background

Mixture-of-Experts. MoE (Shazeer et al., 2017; Cai et al., 2025) is a neural network architecture that scales model capacity by sparsely activating a subset of experts per input. Compared to dense feed-forward layers, MoE layers significantly increase parameter count and representational capacity while keeping the per-token computation bounded (Chen et al., 2022; Chi et al., 2022).

In a typical MoE layer, a router computes routing scores over all experts based on the token's hidden representation, and selects the top-k experts for execution (Liu et al., 2024; Fedus et al., 2022). The selected experts process the input in parallel, and their outputs are aggregated using the routing weights. Formally, given an input x, the MoE output is

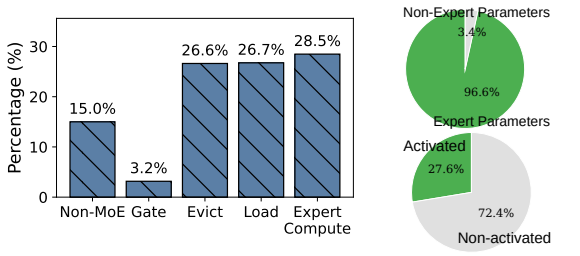
$$y = \sum_{e \in \mathcal{E}_k} G(x)_e \cdot E_e(x), \tag{1}$$

where $\mathcal{E}_k$ denotes the selected expert set, $E_e(\cdot)$ is the e-th expert, and $G(x)_e$ is the corresponding routing weight.

This token-level, router-driven sparse activation is a defining characteristic of MoE architectures. While effective for reducing computation, it induces highly skewed and input-dependent expert access patterns during inference: different tokens may activate entirely different expert subsets across layers, resulting in irregular and rapidly changing memory access behavior (Yao et al., 2024).

Expert Offloading. In MoE models, expert parameters account for the majority of the model size. For example, Mixtral-8×7B contains approximately 46.7B parameters (Jiang et al., 2024a), over 90% of which reside in the expert feed-forward networks, while non-expert parameters constitute only a small fraction of the total. During inference, each token activates roughly 12.9B parameters.

Under memory-constrained settings, such as single-GPU or edge deployments, it is often impractical to keep the full expert pool resident in GPU memory. As a result, prior systems adopt expert offloading (Eliseev and Mazur, 2023), where non-expert parameters and a subset of experts are kept in GPU memory, while the remaining experts are placed in CPU memory or external storage. When required experts are not resident in GPU memory, the system must evict some existing experts to free space and fetch the missing ones from lower-tier storage before computation can proceed.



(a) Inference time breakdown. (b) MoE Parameters.

Figure 2: Inference Latency and Parameter Composition of Mixtral-8×7B. The left panel shows the breakdown of inference time across different execution stages, while the right panel characterizes the composition and utilization of model parameters during inference.

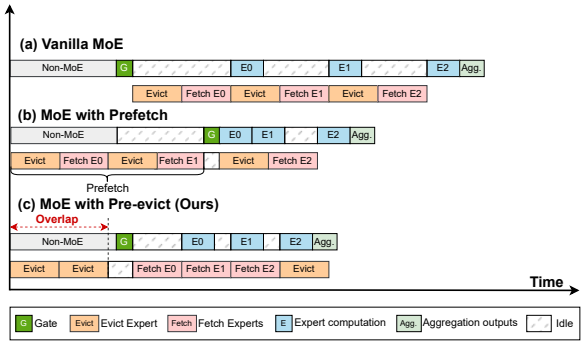


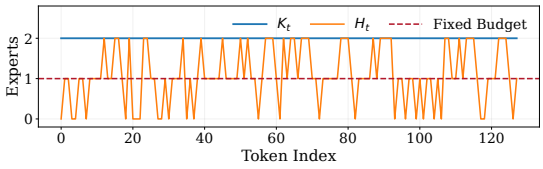
Figure 3: Comparison of MoE offloading pipelines: (a) vanilla on-demand replacement, (b) expert prefetching, and (c) our pre-eviction design that decouples eviction from fetching.

2.2 Motivation

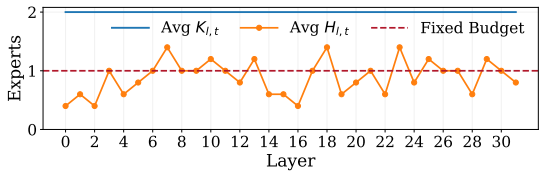
Observation 1: Expert loading dominates MoE inference latency under offloading. Although MoE reduces per-token computation via sparse activation, inference with expert offloading is frequently constrained by data movement rather than computation. When required experts are not resident in GPU memory, the system must evict existing experts and fetch missing ones from lower-tier storage before execution can proceed.

Figure 2(a) shows the latency breakdown of a Mixtral-8×7B layer, where expert eviction and fetching together account for a substantial fraction of the end-to-end latency. Due to the limited bandwidth of host–device interconnects relative to GPU compute throughput, this replacement phase often dominates inference, rendering MoE inference I/O-bound under expert offloading.

Observation 2: MoE offloading pipelines tightly couple expert eviction and fetching. Most existing MoE offloading systems adopt an on-demand scheduling paradigm. After routing decisions are made for each token, the system evicts a subset of resident experts to reclaim GPU memory and then fetches the required experts from lower-tier storage before computation can begin.



(a) Token-wise cache hits at Layer 10.



(b) Average cache hits across consecutive tokens.

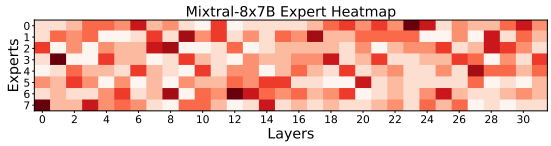
Figure 4: Cache hits under a fixed expert budget.

As illustrated in Fig. 3(a), expert replacement is performed as a serialized "evict-then-fetch" operation, exposing a contiguous I/O-critical region on the post-routing execution path. Since expert loading lies on the critical path, opportunities to overlap data transfer with computation are limited, leading to underutilization of GPU compute resources.

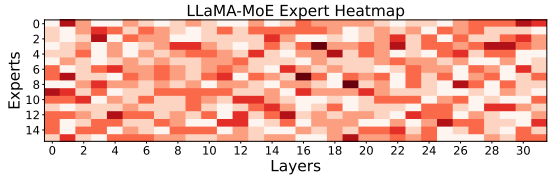
Observation 3: Fixed expert budgets mismatch dynamic routing demand. Most existing MoE offloading systems assign a fixed expert budget B to each layer, implicitly assuming that a static cache capacity can adequately accommodate routing demand during inference. However, analysis of cache hits and expert activations from real inference traces reveals a systematic mismatch between this assumption and actual behavior.

As shown in Fig. 4, the cache hit cardinality $|\mathcal{A}_{l,t} \cap \mathcal{C}_{l,t}|$ varies substantially across both tokens and network depths. This variability persists even when averaged over local token windows, indicating that it reflects an inherent property of MoE inference rather than transient noise.

Under such dynamics, a fixed expert budget induces two predictable regimes. When B is over-provisioned, GPU memory remains persistently underutilized, resulting in wasted capacity. When B is under-provisioned, frequent evictions and fetches become unavoidable, exposing I/O-dominated replacement phases and causing the system to regress toward on-demand swapping behavior. The coexistence of these regimes during inference highlights a structural conflict between static expert budgets and



(a) Expert activation heatmap of Mixtral-8×7B.



(b) Expert activation heatmap of LLaMA-MoE-3.5B.

Figure 5: The expert heat maps of Mixtral-8×7B and LLaMA-MoE-3.5B.

inherently dynamic routing demand, rather than an artifact of specific cache policies or implementations.

Insight: MoE experts exhibit long-tailed popularity and cross-layer consistency. Despite the dynamic nature of routing, expert access during inference exhibits strong short-term locality. As shown in Fig. 5, activation frequencies in Mixtral-8×7B and LLaMA-MoE-3.5B reveal pronounced long-tailed distributions within local token windows, where a small subset of experts is repeatedly activated across adjacent layers and tokens. This behavior arises from the locality of inference: within contiguous input segments, semantic focus evolves gradually, leading to repeated routing decisions over short horizons.

Moreover, due to the residual structure of Transformers (Dalvi et al., 2020; Jiang et al., 2024b), hidden representations across neighboring layers remain highly correlated, inducing cross-layer consistency in expert selection. These properties suggest that recent expert activity provides a reliable short-term signal for expert reuse, enabling informed eviction decisions without speculative prediction.

3 Method

To address the I/O bottlenecks in memory-constrained MoE inference, we propose a framework that proactively manages expert residency *before routing*. Our approach reduces pipeline stalls by separating expert eviction from loading and dynamically adapting expert activation to available GPU memory. The method consists of two components: (i) *Prediction-guided pre-eviction*, which identifies low-utility or infrequently accessed experts using both local hotness and cross-layer routing signals, evicting them proactively to free GPU memory. (ii) *Budget-aware dynamic top-k routing*, which adjusts the set of activated experts at each token based on the memory reclaimed through pre-eviction, effectively transforming a static cache budget into a dynamic effective budget.

3.1 Problem Formulation

We consider an MoE model with L layers, where each layer l contains N experts $\{E_{l,1}, \ldots, E_{l,N}\}$. Given the hidden representation $h_l \in \mathbb{R}^d$ of an input token, the router produces a probability distribution over experts:

$$p_l(h_l) = \mathrm{Softmax}(W_l h_l) \in \mathbb{R}^N. \tag{2}$$

Under top-k routing, the router selects

$$R_l = \mathrm{TopK}(p_l, k), \quad |R_l| = k, \tag{3}$$

and the MoE layer output is

$$y_l = \sum_{i \in R_l} \tilde{p}_{l,i} \cdot E_{l,i}(h_l), \quad \tilde{p}_{l,i} = \frac{p_{l,i}}{\sum_{j \in R_l} p_{l,j}}. \tag{4}$$

Memory-Constrained Inference Setting. In resource-limited settings, only a subset of expert parameters can reside in GPU memory. Let $C_l \subseteq \{1, \ldots, N\}$ denote the set of experts resident in GPU memory before executing layer l, with size constrained by:

$$|C_l| \leq B_l, \tag{5}$$

where B_l is the maximum number of resident experts at layer l.

If $R_l \nsubseteq C_l$, eviction and loading of experts are required, introducing data transfer overhead and I/O latency. Existing MoE offloading methods typically resolve this conflict after routing, exposing eviction and loading on the inference critical path.

Dynamic Budget. Our key question is whether expert residency can be proactively managed *before routing* to better match memory constraints with routing demand. Specifically, before layer l, the algorithm may evict a subset of resident experts:

$$D_l \subseteq C_l, \tag{6}$$

resulting in an updated resident set

$$C'_l = C_l \setminus D_l. \tag{7}$$

This preemptive eviction effectively converts a fixed memory budget into a dynamic, algorithmically controlled budget.

Objective. Our goal is to jointly decide which experts to evict before routing and how to adjust the routing width under the resulting memory budget, so as to minimize expert swapping and I/O overhead while maintaining the model's expressive capacity. Importantly, all expert activations are still determined by the routers based on the true input; our method only regulates memory residency and routing budget, without introducing errors into the computation.

3.2 Expert Importance Modeling

Effective expert management requires quantifying the relative importance of experts in the current and near-future inference steps (Lu et al., 2024). Such importance scores guide pre-eviction and memory budget allocation, ensuring that frequently used or likely-to-be-activated experts remain in GPU memory.

Historical expert hotness. We first model expert importance at a local temporal scale using historical usage statistics. For each expert e, we track recent activation frequency within a sliding window of T tokens. Let $\mathbb{I}_t(e)$ be an indicator for whether e is selected at token t. We define a time-decayed usage intensity:

$$u(e) = \sum_{t=1}^{T} \gamma^{T-t}\, \mathbb{I}_t(e), \tag{8}$$

where $\gamma \in (0, 1]$ controls the relative weight of past activations. Larger γ emphasizes longer-term trends, while smaller γ focuses on recent activity.

We normalize within the resident set C_l to obtain the historical hotness score:

$$h(e) = \frac{u(e)}{\sum_{e' \in C_l} u(e')}. \tag{9}$$

This score reflects relative importance and prevents low-utility experts from remaining in memory, mitigating frequent expert swapping.

Cross-layer routing prediction. While historical hotness provides a stable local signal, it is inherently backward-looking and cannot anticipate upcoming routing changes. To incorporate short-term foresight, we leverage cross-layer correlation in Transformers: hidden states evolve smoothly across adjacent layers, inducing correlated routing behavior. Before layer $l{+}1$, we use hidden states $z_{l,t}$ from layer l to predict routing probabilities via the router g_{l+1}:

$$\hat{r}_{l+1,t} = g_{l+1}(z_{l,t}), \tag{10}$$

$$\hat{\pi}_{l+1,t} = \mathrm{Softmax}(\hat{r}_{l+1,t}), \tag{11}$$

where $\hat{\pi}_{l+1,t}(e)$ represents the predicted assignment probability for expert e. Averaging over tokens yields a predicted importance distribution for layer $l{+}1$:

$$\hat{p}_{l+1}(e) = \frac{1}{T} \sum_{t=1}^{T} \hat{\pi}_{l+1,t}(e). \tag{12}$$

This distribution provides a short-term prior on expert relevance in the next layer and is used to guide pre-eviction and memory release.

Combined importance score. We combine time-decayed historical hotness and cross-layer prediction to construct a unified importance score for eviction ranking. For expert e at layer l, we define

$$s_l(e) = \alpha \cdot h(e) + (1 - \alpha) \cdot \hat{p}_{l+1}(e), \quad \alpha \in [0, 1]. \tag{13}$$

The parameter α balances stable historical information and predictive foresight. The resulting score is used to rank experts for proactive eviction and dynamic memory budgeting, without introducing errors into the router's actual computation path.

3.3 Prediction-Guided Pre-Eviction

The goal of pre-eviction is to release sufficient memory capacity before entering layer $l{+}1$, enabling future expert activations without blocking on I/O. If k experts will be activated at the next layer and some are already predicted to reside in memory, only the remaining experts require eviction. However, since cross-layer predictions are imperfect, we account for uncertainty in memory release.

Let the predicted expert probabilities at layer $l{+}1$ be sorted as $p_{(1)} \geq p_{(2)} \geq \cdots$. For a candidate routing width k, the total memory capacity that must be released before layer $l{+}1$ can be naturally decomposed into two components: (i) a *prediction gap*, representing the minimal additional capacity required to accommodate newly activated experts under a *ground-truth prediction*, and (ii) an *uncertainty redundancy*, accounting for possible prediction errors near the top-k boundary.

Prediction gap. Under perfect prediction, the amount of memory that must be released equals the routing width k minus the number of experts

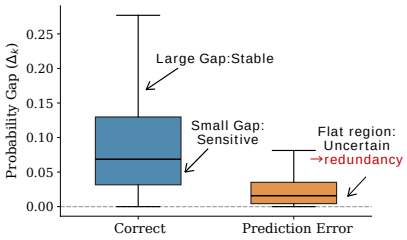


Figure 6: Distribution of top-k probability gaps δ_k under correct and erroneous predictions.

that are predicted to be already resident in memory. Formally, let $\widehat{H}_l(k)$ denote the number of experts among the predicted top-k at layer $l{+}1$ that are currently resident. The prediction gap is then given by

$$k - \widehat{H}_l(k), \tag{14}$$

which represents the minimal capacity required to accommodate newly activated experts.

Boundary uncertainty correction. Prediction errors often occur near the top-k boundary, where probabilities are relatively flat. As shown in Fig. 6, large gaps at the boundary indicate stable routing decisions, while small gaps imply sensitivity to perturbations and higher risk of misprediction.

We use the top-k boundary gap

$$\delta_k = p_{(k)} - p_{(k+1)} \tag{15}$$

as a proxy for prediction uncertainty: large gaps indicate stable routing decisions, while small gaps imply higher sensitivity to perturbations and potential mispredictions. To account for extended flat regions, we introduce a redundancy term:

$$r_k = \sum_{j=0}^{r_{\max}-1} \mathbb{I}\big[\, p_{(k+j)} - p_{(k+j+1)} < \tau \,\big], \tag{16}$$

where τ is a threshold and $r_{\max}$ bounds the maximum redundancy.

Release target. We combine the prediction gap and boundary uncertainty to compute the total capacity to release before layer $l{+}1$:

$$\Delta_l(k) = \big(k - \widehat{H}_l(k)\big) + r_k, \tag{17}$$

Pre-eviction proceeds by evicting experts in ascending order of importance $s_l(e)$ until at least $\Delta_l(k)$ slots are freed.

3.4 Budget-Aware Dynamic Top-k Routing

After pre-eviction and before routing at layer l, let $\mathcal{C}_l$ denote the resident expert set, with size $H_l = |\mathcal{C}_l|$ and total capacity B. The number of available slots for new experts is

$$C_l^{\text{avail}} = B - H_l. \tag{18}$$

Let experts be sorted by routing probability as $\{e_{(1)}, e_{(2)}, \ldots, e_{(E)}\}$. For a candidate routing width k, the number of experts that must be newly loaded is

$$u_l(k) = \big|\{\, e_{(j)} \notin \mathcal{C}_l \mid 1 \le j \le k \,\}\big|. \tag{19}$$

This quantity measures the number of cache misses induced by top-k routing under the current resident set.

The memory constraint requires $u_l(k) \le C_l^{\text{avail}}$. We therefore select the maximum feasible routing width

$$k_l = \max\big\{\, k \,\big|\, 1 \le k \le k_{\max},\ u_l(k) \le C_l^{\text{avail}} \,\big\}, \tag{20}$$

and activate

$$\mathcal{R}_l = \{e_{(1)}, e_{(2)}, \ldots, e_{(k_l)}\}. \tag{21}$$

This mechanism ensures that routing decisions remain memory-feasible, while prioritizing resident experts and limiting additional expert loading.

4 Experiments

4.1 Experimental Setup

Hardware. All experiments are conducted on a server equipped with an NVIDIA H20 GPU with 140 GB HBM memory and 78 streaming multiprocessors (SMs). The system uses an Intel Xeon 6759P-C CPU with 240 cores and 1.9 TiB of host memory. The CPU and GPU are connected via PCIe 4.0. We measure the effective CPU–GPU bandwidth using large contiguous tensor transfers, obtaining 50.45 GB/s for host-to-device and 47.46 GB/s for device-to-host communication. The storage subsystem consists of a PCIe 4.0 $\times$4 NVMe SSD (Intel SSDPF2KX038T1, 3.5 TB), with peak sequential read and write bandwidths of approximately 7.0 GB/s and 4.2 GB/s, respectively.

Models and Datasets. We evaluate the proposed method on two representative Mixture-of-Experts language models: Mixtral-8$\times$7B (Jiang et al., 2024a) and LLaMA-MoE-3.5B (Zhu et al., 2024). As summarized in Table 1, Mixtral-8$\times$7B contains

Table 1: Configuration of two evaluated MoE models.

	Mixtral-8×7B	**LLaMA-MoE-3.5B**
Total Parameters	46.7B	6.7B
Activated Params	13B	3.5B
Expert Parameter Ratio	96%	64%
Expert Number	8	16
Top-K	2	4
Layers	32	32

Table 2: **Fetch accuracy under expert offloading.** Mix/LLA denote Mixtral and LLaMA, respectively; P/R indicate precision and recall.

Method	Mix-P	Mix-R	LLA-P	LLA-R
MoE-Offload	85.6%	84.4%	85.7%	77.8%
MoE-APEX	88.9%	90.9%	88.6%	93.4%
Ours	100%	100%	100%	100%

8 experts per layer and activates 2 experts per token during inference. LLaMA-MoE-3.5B is built upon a 6.7B-parameter LLaMA-2-7B backbone, introducing 16 experts per layer and adopting a Top-4 routing strategy. We use the Alpaca instruction dataset (Taori et al., 2023) to construct short inputs and medium-length inputs, and select long-context samples from LongBench (Bai et al., 2024) to emulate higher context lengths.

Baselines. We compare against three offloading-based baselines that differ in their expert loading strategies. (1) **On-demand Offloading** (Eliseev and Mazur, 2023): a deterministic expert offloading baseline that performs on-demand loading without any prefetching. (2) **MoE-Offloading** (Eliseev and Mazur, 2023): an expert offloading baseline that augments on-demand loading with lightweight speculative prefetching to partially overlap communication and computation. (3) **MoE-APEX** (Tang et al., 2026): an expert offloading system that performs more aggressive, router-guided speculative prefetching across layers to maximize compute–I/O overlap.

4.2 Correctness of Post-Gate Expert Loading

To validate the correctness of our pre-eviction with post-gate loading mechanism, we evaluate *fetch accuracy*, which measures whether the experts fetched from CPU or external storage exactly match those selected by the router after gating.

Concretely, at each layer and decoding step, we compare the set of fetched experts with the ground-truth post-gate activation set and compute fetch precision and recall. All metrics are averaged across layers and tokens.

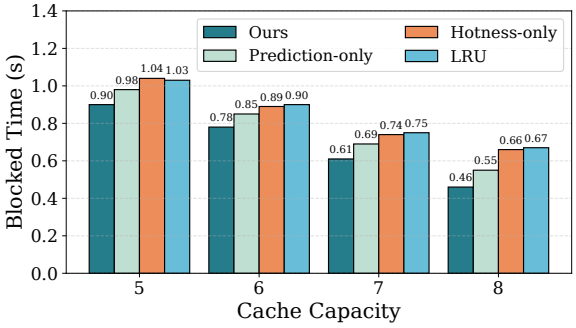


Figure 7: Blocking I/O time across different cache capacities for various eviction strategies.

As shown in Table 2, prefetch-based methods may incur redundant or delayed transfers due to speculative loading and mispredictions. In contrast, our approach deterministically triggers expert loading only after gate decisions are finalized, achieving perfect fetch precision and recall by design.

4.3 Blocking I/O Reduction and Transfer Efficiency

Table 3 presents the end-to-end I/O behavior of different expert scheduling strategies under memory-constrained inference. For both Mixtral-8×7B and LLaMA-MoE models, our method consistently achieves the lowest *blocking I/O time*, reducing post-gate stalls by 31%–39% on Mixtral and 40%–46% on LLaMA compared to prefetch-based baselines. These improvements are achieved without increasing the overall data transfer volume. Prefetch-based approaches, such as Mixtral-style and MoE-APEX, attempt to overlap computation with I/O through speculative loading, but this often results in 12%–20% redundant transfers and higher total I/O volume. In contrast, importance-aware pre-eviction proactively frees memory before routing, shortening the post-gate I/O-critical region and enabling more effective overlap between computation and I/O without unnecessary data movement.

4.4 Ablation Study

Expert importance scoring. We evaluate alternative eviction heuristics by replacing the proposed importance score with prediction-only, hotness-only, and LRU policies. Across all cache capacities, these variants incur higher blocking I/O time than the full method (Fig. 7). In particular, the full method reduces blocking I/O by 8%–16% compared to prediction-only, and by 13%–32% relative to hotness-only and LRU. These results indicate

Table 3: Expert scheduling metrics. **Blocking I/O Time**: the portion of I/O latency that cannot be overlapped with computation and directly contributes to end-to-end inference latency. **Total I/O Time**: the total data transfer time on the critical path, reflecting overall bandwidth consumption. **Transfer Volume**: the amount of parameters transferred per token, measuring I/O pressure. **Redundant Transfer**: avoidable data movement, including fetched but unused experts and extra transfers caused by premature eviction.

		Blocking I/O Time				**Total I/O Time**			
Model	**Dataset**	Offload	Mixtral	MoE-APEX	**Ours**	Offload	Mixtral	MoE-APEX	**Ours**
Mixtral	Alpaca	44.35	39.62	35.43	**24.31**	44.35	55.99	50.84	**43.67**
	LongBench	89.17	82.17	70.53	**49.96**	89.17	114.92	100.67	**88.69**
LLaMA	Alpaca	2.13	1.44	1.50	**0.81**	2.13	2.83	3.10	**1.62**
	LongBench	9.18	6.39	6.11	**3.63**	9.18	12.28	12.62	**7.20**

		Transfer Volume (GB/token)				**Redundant Transfer**			
Model	**Dataset**	Offload	Mixtral	MoE-APEX	**Ours**	Offload	Mixtral	MoE-APEX	**Ours**
Mixtral	Alpaca	9.82	12.39	11.25	**9.66**	**0%**	15.51%	19.59%	3.41%
	LongBench	9.87	12.72	11.14	**9.81**	**0%**	16.30%	20.63%	3.24%
LLaMA	Alpaca	0.89	1.18	1.29	**0.68**	**0%**	12.82%	14.88%	4.03%
	LongBench	0.96	1.28	1.31	**0.75**	**0%**	13.97%	16.18%	4.00%

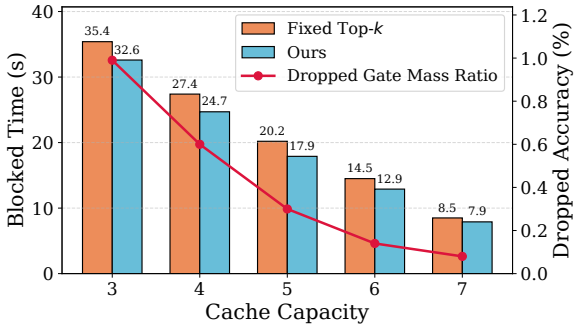


Figure 8: Comparison of budget-aware dynamic top-k and fixed top-k routing across cache capacities.

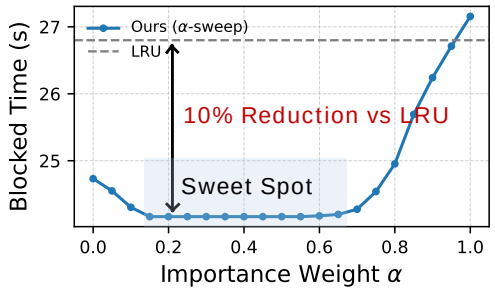


Figure 9: Sensitivity of blocked I/O time to the importance weight α.

that combining short-term predictive signals with historical usage is essential for robust and stable pre-eviction.

Budget-aware dynamic top-k. Figure 8 compares budget-aware dynamic top-k routing with fixed top-k under varying memory capacities. Dynamic top-k reduces post-gate expert loading and lowers blocking I/O by 7%–13% in memory-constrained regimes, while incurring less than 1% degradation in the accuracy proxy even at the tightest capacity. These results indicate that budget-aware routing mainly suppresses low-probability experts near the routing boundary, thereby achieving substantial I/O savings with minimal impact on routing fidelity.

Sensitivity Analysis. Figure 9 shows the sensitivity of blocking I/O time to the importance weight α. Our method consistently outperforms LRU across a wide range of α, forming a broad performance plateau and indicating low sensitivity to hyperparameter tuning. When α is too small, noisy prediction signals dominate eviction, while as $\alpha \to 1$, the policy reduces to a hotness-only strategy similar to LRU. This behavior is expected and confirms LRU as a limiting case of our formulation.

5 Conclusion

We present ACTIVEEVICT, a budget-aware pre-eviction approach for efficient MoE offloading inference. By proactively releasing GPU memory before routing, ACTIVEEVICT decouples expert eviction from loading and enables a dynamically induced effective cache budget. Across multiple MoE models, this design reduces blocking I/O time by 31%–46% compared to prefetch-based baselines, while incurring less than 1% accuracy degradation, demonstrating that proactive eviction is an effective alternative to speculative prefetching for alleviating I/O bottlenecks in MoE inference.

Limitations

This work focuses on improving MoE inference efficiency under memory-constrained deployment settings where expert offloading is required. Accordingly, the proposed pre-eviction and budget-aware routing mechanisms are designed for inference-time execution and do not address MoE training or fine-tuning, where routing behavior and memory dynamics differ substantially. In addition, our approach relies on the empirical observation that expert activations exhibit short-term locality and cross-layer consistency during typical decoding workloads. While this assumption holds for the evaluated models and datasets, workloads with highly irregular routing patterns or abrupt semantic shifts may reduce the effectiveness of importance estimation and proactive eviction.

We acknowledge these limitations and leave the extension of our framework to broader training settings and more diverse inference workloads for future work.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. 2022. Towards understanding the mixture-of-experts layer in deep learning. *Advances in neural information processing systems*, 35:23049–23062.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, and 1 others. 2022. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, and 1 others. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pages 5547–5569. PMLR.

Artyom Eliseev and Denis Mazur. 2023. Fast inference of mixture-of-experts language models with offloading. *arXiv preprint arXiv:2312.17238*.

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

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. 2024. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference. in 2024 acm/ieee 51st annual international symposium on computer architecture (isca). *IEEE*, 2:1018–1031.

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, and 1 others. 2024a. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. 2024b. Tracing representation progression: Analyzing and enhancing layer-wise similarity. *arXiv preprint arXiv:2406.14479*.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. 2024. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models. *arXiv preprint arXiv:2402.07033*.

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

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624.

N Shazeer, A Mirhoseini, K Maziarz, A Davis, Q Le, G Hinton, and J Dean. 2017. The sparsely-gated mixture-of-experts layer. *Outrageously large neural networks*, 2.

Peng Tang, Jiacheng Liu, Xiaofeng Hou, Yifei Pu, Jing Wang, Pheng-Ann Heng, Chao Li, and minyi Guo. 2026. Moe-apex: An efficient moe inference system with adaptive precision expert offloading. In *International Conference on Architectural Support for Programming Languages and Operating Systems*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yuanxin Wei, Jiangsu Du, Jiazhi Jiang, Xiao Shi, Xianwei Zhang, Dan Huang, Nong Xiao, and Yutong Lu. 2024. Aptmoe: Affinity-aware pipeline tuning for moe models on bandwidth-constrained gpu nodes. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. 2024. Moe-infinity: Activation-aware expert offloading for efficient moe serving. *arXiv preprint arXiv:2401.14361*, 3.

Jinghan Yao, Quentin Anthony, Aamir Shafi, Hari Subramoni, and Dhabaleswar K DK Panda. 2024. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 915–925. IEEE.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*.