CommitMoE: Efficient Fallback-Free MoE Inference with Offloading Under GPU Memory Constraints

Han Li, Jingwei Sun*, Junqing Lin, Guangzhong Sun*

University of Science and Technology of China
hanli06@mail.ustc.edu.cn, sunjw@ustc.edu.cn, linjunqing@mail.ustc.edu.cn, gzsun@ustc.edu.cn

Abstract

Mixture of Experts (MoE) models have emerged as a promising approach to scale language models efficiently by activating only a subset of parameters for each input. However, deploying these models under GPU memory constraints remains challenging, as existing offloading strategies incur significant overhead from CPU-GPU data transfers. While prior work has explored prefetching techniques to mitigate this bottleneck, these methods require costly fallback mechanisms when predictions fail. Since expert transfers cannot be canceled once initiated, the correct experts need to be loaded on demand sequentially, introducing additional latency. To address this, we present CommitMoE, a novel approach featuring a Commit Router that makes execution decisions based on expert predictions without fallback mechanisms. Our key insight reveals that router certainty strongly correlates with prediction accuracy, while in low-certainty scenarios, the model output demonstrates inherent robustness to expert selection. Leveraging this insight to design a systems-level solution, CommitMoE achieves 1.3× to 9.4× faster inference across different environments and datasets compared to state-of-the-art offloading frameworks while maintaining model quality.

1 Introduction

The rapid advancement of Large Language Models (LLMs) (Anthropic 2024; Team et al. 2024; Guo et al. 2025) has revolutionized artificial intelligence, exhibiting unprecedented capabilities across various tasks. However, training and deploying such massive models present significant computational and resource challenges. The Mixture of Experts (MoE) framework has emerged as a breakthrough architecture that enables significant expansion of model capacity while maintaining computational efficiency (Dai et al. 2024; Liu et al. 2024; Jiang et al. 2024; Yang et al. 2024). By selectively activating specialized "expert" subnetworks for each input, MoE models utilize only a fraction of their total parameters during inference, substantially reducing computational load without compromising powerful performance.

Despite these advantages, MoE models remain difficult to deploy in GPU memory-constrained environments where the overall parameter size often exceeds the available GPU memory capacity. This challenge is particularly pronounced in resource-limited scenarios such as edge deployments, where local inference capabilities are required without relying on distributed computing infrastructure. To alleviate the GPU memory pressure, inactive expert parameters can be offloaded to higher-capacity, lower-bandwidth memory tiers such as CPU RAM or NVMe storage (Cui et al. 2023; Eliseev and Mazur 2023; Hwang et al. 2024; Rajbhandari et al. 2022; Zhong et al. 2024). During inference, required expert parameters are dynamically reloaded onto the GPU, leading to substantial data transfer latency. This data transfer overhead, as shown in Figure 1, accounts for the major inference time, hindering the practical application of large MoE models in latency-sensitive scenarios.

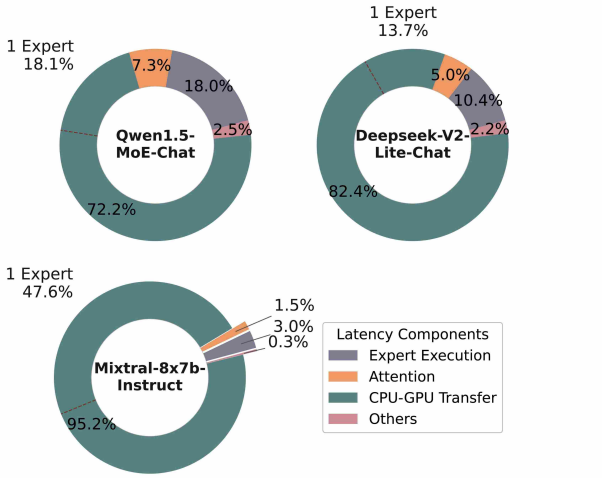

Figure 1: Breakdown of per-MoE-block inference latency for three typical MoE models (with 2, 4, and 6 experts activated per token, respectively) on NVIDIA GeForce RTX 4090 using on-demand loading. CPU-GPU data transfer dominates the total latency.

Prefetching techniques have been proposed to mitigate data transfer latency by overlapping expert loading with non-expert computation on the GPU (Eliseev and Mazur 2023; Xue et al. 2025; Zhong et al. 2024; Song et al. 2025). Given the dynamic nature of expert selection, prefetching decisions are typically guided by predicted activation prob-

*Corresponding authors.

abilities, obtained via routers in prefetched layers (Eliseev and Mazur 2023), predictive models (Song et al. 2025), or historical routing statistics (Xue et al. 2025). Experts with high predicted activation likelihoods are prefetched to reduce inference latency. However, when the prefetched experts are incorrect, the system falls back to on-demand loading, discarding the mispredicted experts. Such mispredictions not only eliminate the opportunity for overlap but also introduce additional latency, as CUDA's asynchronous memory copy semantics require the GPU to complete the incorrect transfer before initiating the correct one. For models with different expert activation patterns (as shown in Figure 1), each expert misprediction can introduce an additional 13.7%-47.6% overhead due to the sequential nature of expert parameter transfers, thus diminishing the achievable speedup gains. Pre-gated MoE (Hwang et al. 2024) restructures the model architecture to enable prefetching by having each layer's router determine the expert selection for the next layer. Although this approach improves execution efficiency, it requires a complete retraining of the model.

The limitations of prefetching highlight the need for a routing strategy that (i) improves decision accuracy, (ii) minimizes the runtime impact of mispredictions, and (iii) maintains compatibility with existing pre-trained models without requiring substantial architectural modifications. To explore the characteristics of MoE routing mechanisms, we analyze router behavior using two typical MoE models on mathematical reasoning tasks. Our analysis reveals significant variability in routing certainty: some inputs are assigned to experts with high certainty, while others yield ambiguous selections. Notably, the accuracy of the expert activation predictor is strongly correlated with certainty, where high-certainty inputs tend to produce more accurate outputs, whereas low-certainty inputs exhibit degraded accuracy. Importantly, under low-certainty conditions, model quality remains stable across different expert selections, indicating a degree of robustness to routing variations. These observations suggest that discarding mispredicted experts may be unnecessary when routing certainty is low. This motivates a more fundamental shift: eliminating fallback mechanisms and directly executing the predicted experts, regardless of their alignment with the router's original selection.

Based on this insight, we present **CommitMoE**, an efficient MoE inference framework featuring a **Commit Router**, which eliminates fallback mechanisms by unconditionally adopting predicted expert selections. Specifically, the Commit Router employs a lightweight predictor to estimate expert usage and commits to the predicted experts. The predicted experts replace the actual activation experts for token computation, significantly reducing data transfer overhead. CommitMoE leverages the model's inherent robustness in low-certainty scenarios, avoiding the costs of fallback execution without requiring model retraining. Furthermore, to address potential quality loss from suboptimal expert selection, CommitMoE introduces an Output Weight Adjustment (OWA) mechanism that compensates for discrepancies between committed and ideal routes. Our main contributions are as follows:

- We identify and empirically validate the relationship between MoE router certainty, expert predictability, and model output robustness, providing a new perspective for optimizing MoE inference.
- We propose the Commit Router, a mechanism that makes fallback-free expert selection decisions directly based on the prediction results, thus eliminating unnecessary loading of experts.
- We present CommitMoE, an efficient MoE inference framework that integrates the Commit Router and OWA with an offloading strategy, significantly reducing inference latency under GPU memory constraints.
- We conduct extensive evaluations demonstrating that CommitMoE achieves significant speedups compared to state-of-the-art offloading frameworks while maintaining model quality.

2 Background and Related Work

2.1 MoE Architecture

The Mixture of Experts (MoE) architecture enhances large language models by replacing dense feed-forward networks with a set of N specialized "expert" subnetworks $E_1, E_2, \ldots, E_N$ (Shazeer et al. 2017). For each input token representation x, a router G dynamically determines how to combine these experts. The router computes gating scores $G(x) = (G_1(x), G_2(x), \ldots, G_N(x))$ by applying a softmax function to the logits obtained from a linear transformation of the input x with a learnable weight matrix W_g:

$$G(x) = \text{softmax}(W_g x). \tag{1}$$

Based on these scores, the router selects a small subset of k experts (where $k \leq N$), typically those corresponding to the top k scores. Let $I_{topk}(x) \subseteq \{1, \ldots, N\}$ be the set of indices of these selected experts for input x. The final output y of the MoE layer is then computed as a weighted sum of the outputs of these selected experts:

$$y = \sum_{i \in I_{topk}(x)} G_i(x) E_i(x). \tag{2}$$

2.2 Related Work

Recent years have witnessed the emergence of various approaches to address the challenges of deploying MoE models under GPU memory constraints. These approaches can be broadly categorized into model compression and expert offloading strategies. Model compression techniques such as quantization (Eliseev and Mazur 2023; Yi et al. 2025; Tang et al. 2024; He et al. 2025; Imani, Amirany, and El-Ghazawi 2024; Frantar and Alistarh 2023) and pruning (Lu et al. 2024; Xie et al. 2024; Lu et al. 2025) reduce memory requirements by decreasing parameter precision or eliminating redundant parameters. While effective at reducing memory footprint, these methods often incur significant quality degradation, particularly at higher compression rates, limiting their applicability in scenarios demanding high model fidelity. Expert offloading strategies, on the other hand, maintain model quality by dynamically managing expert parameters across memory hierarchies (Cui et al. 2023; Rajbhandari

et al. 2022; Zhong et al. 2024; Xue et al. 2025; Cao et al. 2025; Wei et al. 2024; Yuan, Ma, and Talati 2025; Sheng et al. 2023).

The optimization of offloading strategies have evolved along two primary dimensions: throughput (Cao et al. 2025; Wei et al. 2024; Yuan, Ma, and Talati 2025; Sheng et al. 2023) and latency (Cui et al. 2023; Rajbhandari et al. 2022; Zhong et al. 2024; Xue et al. 2025). For throughput, MoE-Lighting (Cao et al. 2025) proposes HRM modeling to maximize resource utilization with CGOPIPE. MoE-Lens (Yuan, Ma, and Talati 2025) introduces a holistic performance model that accounts for CPU memory capacity and workload heterogeneity, achieving near-hardware-limit throughput. For latency optimization, which is the primary focus of our work, MoE-Offloading (Eliseev and Mazur 2023) introduced a framework that observes token-wise expert selection similarities and adopts a least recently used (LRU) algorithm to manage the expert cache. MoE-Infinity (Xue et al. 2025) observed correlations in expert selection across layers, using previous layers' activations to predict subsequent expert activations at the request level, while AdapMoE (Zhong et al. 2024) refines prediction at the layer level. Pregated MoE (Hwang et al. 2024) attempts to determine expert selections through architectural modifications, necessitating complete model retraining and compromising model flexibility. Our work, CommitMoE, bridges these gaps with a fallback-free strategy that maintains model quality without retraining, leveraging the insight that routing uncertainty correlates with output resilience across expert selections.

3 Observation and Insight

This section analyzes expert selection patterns during decoding of MoE models to inform efficient offloading strategies. We conducted empirical analysis on the relationship between router certainty and expert predictability using Qwen1.5-MoE-Chat (Team 2024) and DeepSeek-V2-Lite-Chat (Liu et al. 2024) on GSM8K (Cobbe et al. 2021), as mathematical reasoning involves sequential problem-solving steps that create varying computational demands.

Our methodology involved training simple Multi-Layer Perceptron (MLP) predictors: single-layer for same-token predictions and two-layer for cross-token predictions (detailed in Section 4.2). The predictor takes router input at layer i and predicts experts selected at layer $i+1$ (or the first MoE layer of the next token if layer i is the last MoE layer). We quantify router certainty using Kullback-Leibler divergence (KLD) between the router's softmax distribution and a uniform distribution, with higher KLD indicating a more peaked distribution, signifying higher certainty, while a lower KLD suggests the distribution is closer to uniform, indicating lower certainty.

Figure 2 illustrates the relationship between router certainty and prediction accuracy across different layers for both models. The prediction accuracy is defined as:

$$\text{Accuracy} = \left\lfloor \frac{|\mathcal{P} \cap \mathcal{T}|}{k} \times 100\% \right\rceil, \tag{3}$$

where $\mathcal{P}$ represents the set of predicted experts, $\mathcal{T}$ represents the set of true experts selected by the router, k is the number of experts selected per token, and $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. Our analysis reveals several key findings:

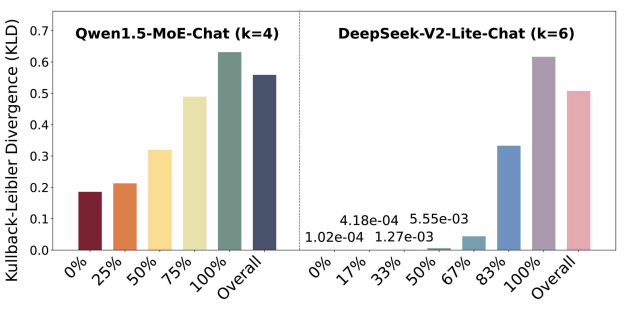

Figure 2: Average Kullback-Leibler Divergence vs. Prediction Accuracy for Qwen1.5-MoE-Chat (k=4) and DeepSeek-V2-Lite-Chat (k=6) on GSM8K dataset. Results are averaged over all tokens across MoE layers during inference.

Finding 1: **Low Certainty Signals Challenging and Potentially Less Critical Predictions.** Instances with the lowest prediction accuracy (0%) consistently show significantly lower average KLD compared to the overall average. This indicates that when router distributions approach uniformity, prediction becomes inherently more challenging and often coincides with reduced model sensitivity to expert choice.

Finding 2: **High Certainty Correlates with High Prediction Accuracy.** Instances with higher prediction accuracy (75%-100%) demonstrate notably higher average KLD, indicating that when routers exhibit strong preferences for specific experts, the predictor more accurately captures these expert selection preferences.

Finding 3: **Monotonic Relationship between Certainty and Accuracy.** We observe a clear positive correlation: as the number of correctly predicted experts increases, the corresponding average KLD increases, reinforcing the strong link between router certainty and prediction feasibility.

Finding 4: **Model Robustness to Expert Selection When Prediction Fails.** For tokens where our predictor failed to predict any expert correctly, we replaced the originally selected experts with prediction-based alternatives. Evaluating on GSM8K (5-shot), we observed negligible changes, and even slight improvements in task accuracy (Table 1). To validate this finding across different task domains, we conducted additional experiments on BBH (Suzgun et al. 2022) (3-shot), which confirmed similar robustness patterns, suggesting that when prediction is difficult, model output is less sensitive to specific expert selection across diverse reasoning tasks.

4 System Design

In this section, we detail CommitMoE, our framework for accelerating MoE inference under memory constraints through fallback-free expert prediction commitment. CommitMoE is systematically designed to optimize the decoding phase of MoE inference.

Model	GSM8K		BBH	
	Original	w/ Replacement	Original	w/ Replacement
Qwen1.5-MoE-Chat	53.22%	54.51%	36.67%	36.75%
DeepSeek-V2-Lite-Chat	69.59%	69.74%	49.12%	49.16%

Table 1: Model Accuracy Comparison: Original Model vs. Replacing Experts in Cases with 0 Correct Predictions.

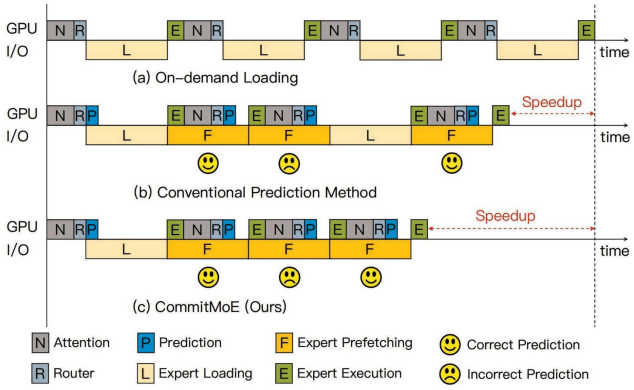

Figure 3: Comparison of inference timelines under offloading: (a) Standard on-demand loading, (b) Prefetching with fallback, and (c) CommitMoE's fallback-free execution. CommitMoE eliminates the additional data transfer overhead incurred when predictions are incorrect.

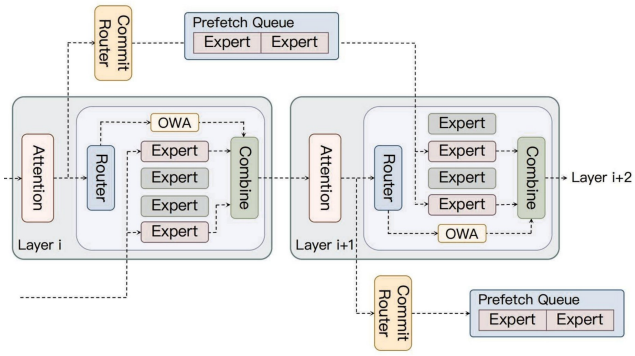

Figure 4: CommitMoE system architecture showing the Commit Router predicting experts for the next layer while the current layer executes, and the Output Weight Adjustment (OWA) mechanism refining expert contributions.

4.1 Overview of CommitMoE

CommitMoE optimizes MoE offloading by fundamentally changing how expert predictions guide execution. While conventional prefetching methods discard predictions upon mismatch with the router's selection, triggering costly on-demand expert loading (Figure 3b), CommitMoE employs an *unconditional commitment* strategy: it always uses the prefetched experts for computation regardless of whether they match the original router's choice. As shown in Figure 3c, this eliminates the additional CPU-GPU data transfer required by prediction fallback.

Figure 4 provides a high-level overview. For the MoE layer i (where $i > 0$), the process begins asynchronously with computation at layer $i-1$. The Commit Router takes the input hidden state for the original router at layer $i-1$ and predicts the expert set $E_{pred}^{(i)}$ likely to be selected at layer i. CommitMoE then initiates asynchronous prefetching of these experts from CPU to GPU memory.

When execution reaches layer i, CommitMoE directly uses the prefetched experts $E_{pred}^{(i)}$. The original router still computes its gating scores $G_{true}^{(i)}$, but these are only used by the Output Weight Adjustment (OWA) mechanism (Section 4.3) to refine expert contributions. Concurrently, the Commit Router predicts experts for layer $i+1$, continuing the pipeline. For the first MoE layer of the first token in the decoding phase, CommitMoE defaults to standard expert selection. For the first MoE layer of subsequent tokens, the expert selection is determined by the Commit Router of the last MoE layer from the previous token.

4.2 Commit Router Predictor

The Commit Router predicts the expert selection of the next MoE layer to enable proactive prefetching. It takes the same input hidden state $h^{(i)}$ as the original router and predicts the probability distribution over experts that the original router at layer $i+1$ is expected to produce. The final set of predicted experts $E_{pred}^{(i+1)}$ is derived by selecting the top k experts from this predicted distribution $P_{pred}^{(i+1)}$.

We implement the Commit Router using lightweight MLPs: a single-layer MLP is sufficient for same-token predictions due to the high cosine similarity between adjacent layers within the same token sequence (Zhong et al. 2024), while a two-layer MLP (with hidden dimension twice the input size) is used for cross-token predictions due to the increased complexity of predicting the first MoE layer of the next token. These predictors are trained offline by minimizing the KL divergence between the softmax of the target logits and the predictor's output:

$$\mathcal{L}_{KL} = D_{KL}\Big(\text{softmax}(R_{true}^{(i+1)}(h^{(i+1)})) \,||\, \text{softmax}(R_{pred}^{(i)}(h^{(i)}))\Big). \tag{4}$$

4.3 Output Weight Adjustment (OWA)

Although CommitMoE's fallback-free strategy optimizes performance, differences between predicted experts $E_{pred}^{(i)}$

and original router-selected experts $E^{(i)}_{true}$ may affect model performance on the downstream task. To mitigate this potential problem, CommitMoE provides an optional Output Weight Adjustment mechanism for scenarios where expert choice matters most (high-router certainty cases identified in Section 3). OWA is only used when prediction accuracy is high but not perfect, as the model may be more sensitive to expert selection in these cases. In the case of lower prediction accuracy, the model exhibits robustness to expert selection and therefore bypasses OWA, relying on the inherent resilience of the model.

When triggered, OWA employs a Redistribution and Scaling approach. Let $Pred$ be the index set for $E^{(i)}_{pred}$, $True$ for $E^{(i)}_{true}$, and Correct_Mask$_k = 1$ if $k \in Pred \cap True$, and 0 otherwise. The adjustment involves:

First, redistributing score mass from missed experts to correctly predicted ones, scaled by a hyperparameter α_1. This step aims to compensate the correctly chosen experts for the contribution of the missed ones:

$$G'_{pred,k} = G^{(i)}_{true,k} \cdot \text{Correct\_Mask}_k \cdot \left(1 + \frac{\sum_{j \in True \setminus Pred} G^{(i)}_{true,j}}{\sum_{l \in Pred} \text{Correct\_Mask}_l \cdot G^{(i)}_{true,l}}\right) \cdot \alpha_1, \tag{5}$$

with $G'_{pred,k} = G^{(i)}_{pred,k}$ when Correct_Mask$_k = 0$.

Second, applying a scaling factor to maintain appropriate magnitude, further adjusted by a hyperparameter α_2. This step aims to maintain an appropriate overall magnitude for the adjusted scores:

$$G''_{pred,k} = G'_{pred,k} \cdot \frac{\sum_{j \in True} G^{(i)}_{true,j}}{\sum_{l \in Pred} G'_{pred,l}} \cdot \alpha_2. \tag{6}$$

These adjusted scores are then used to combine the outputs of the experts executed. Hyperparameters α_1 and α_2 allow empirical tuning across different models.

5 Evaluation

In this section, we present a comprehensive evaluation of CommitMoE across multiple dimensions. We first describe our experimental setup, then analyze CommitMoE's performance in terms of model quality preservation and inference efficiency. We conduct ablation studies for each component of CommitMoE, with results presented alongside the corresponding analysis in their respective subsections.

5.1 Experimental Setup

Implementation. CommitMoE is developed using Python and C++, built on PyTorch (Paszke et al. 2019) and MoE-Infinity (Xue et al. 2025) frameworks, and leverages asynchronous CUDA multi-streaming technology to overlap data movement with computation.

Hardware. Experiments are conducted on two different systems. Environment 1 consists of an NVIDIA RTX 4090 GPU paired with an Intel Xeon W5-3435X CPU. Environment 2 features an NVIDIA GeForce RTX 2080Ti GPU with an Intel Xeon E5-2680 v4 CPU, characterized by comparatively lower computational capability and PCIe bandwidth. Detailed specifications are provided in Table 2.

	Environment 1	**Environment 2**
CPU (Intel Xeon)	W5-3435X	E5-2680 v4
GPU (NVIDIA)	RTX 4090	RTX 2080Ti
GPU Memory	24GB	11GB
PCIe	Gen4 x16 (32GB/s)	Gen3 x16 (16GB/s)

Table 2: Hardware Configurations for Experiments.

Models and Datasets. We evaluate CommitMoE on three representative open-source MoE models: Qwen1.5-MoE-Chat (Team 2024), DeepSeek-V2-Lite-Chat (Liu et al. 2024), and Mixtral-8x7B-Instruct (Jiang et al. 2024). For the Commit Router training, we collect data from the training set of different sources, including GSM8K, 7,000 samples from Alpaca dataset (Taori et al. 2023), 10,000 samples from BigBench (Srivastava et al. 2022), and 12,000 samples from WMT16 German-English translation corpus (Bojar et al. 2016). The training data consists of router hidden states and corresponding expert activation patterns from the first 20 generated tokens when these models process the input prompts. For evaluation, we use benchmarks requiring multi-token generation: GSM8K (Cobbe et al. 2021) for mathematical reasoning, TruthfulQA (Lin, Hilton, and Evans 2022) for factual accuracy and truthfulness, BBH (Suzgun et al. 2022) for logical reasoning and multi-step deduction and WMT16 German-English translation (denoted as WMT16-de-to-en) (Bojar et al. 2016) for cross-lingual understanding. The evaluation code is based on LM Harness framework (Gao et al. 2024). We maintain a strict separation between training and evaluation data to ensure an unbiased assessment of CommitMoE's performance.

Training Details. We train the Commit Router using the Adam (Kingma and Ba 2017) optimizer with an initial learning rate of 5e-5. A step-based learning rate scheduler is applied, reducing the learning rate by a factor of two every 5 epochs. To monitor generalization performance and mitigate overfitting, we split the dataset into training and validation sets with an 80/20 ratio using random seed 42 and apply early stopping based on validation loss.

Baselines. We compare CommitMoE against two state-of-the-art frameworks: **vLLM** (Kwon et al. 2023)(v0.7.0), which employs on-demand loading for MoE with PagedAttention for memory efficiency, and **MoE-Infinity** (Xue et al. 2025), which optimizes MoE offloading via request-level expert prefetching with LFU caching. Both represent leading approaches for memory-constrained MoE inference.

5.2 Model Quality Evaluation

A primary concern when modifying inference procedures is maintaining model quality. For our CommitMoE implementation, the OWA mechanism was configured with model-specific hyperparameters: $\alpha_1 = 1.15$ and $\alpha_2 = 1.00$ for DeepSeek-V2-Lite-Chat, $\alpha_1 = 1.30$ and $\alpha_2 = 1.00$ for Qwen1.5-MoE-Chat, and $\alpha_1 = 1.20$ and $\alpha_2 = 1.06$ for

Model	Variant	GSM8K	TruthfulQA		BBH	WMT16-de-to-en	Average
			R1[1]	R2[1]			
Mixtral-8x7B-Instruct	Base	*59.96*	*54.10*	*45.65*	*59.28*	*35.25*	*50.85*
	CommitMoE(w/o OWA)	59.59	54.96	45.04	59.45	36.07	51.02
	CommitMoE(w Neutral OWA[2])	58.15	54.10	44.92	59.47	35.62	50.45
	CommitMoE(w OWA)	58.37	54.95	44.92	59.71	35.69	50.73
Qwen1.5-MoE-Chat	Base	*53.22*	*51.41*	*38.43*	*36.67*	*36.10*	*43.17*
	CommitMoE(w/o OWA)	50.34	51.29	38.19	36.03	36.07	42.38
	CommitMoE(w Neutral OWA[2])	51.63	51.28	37.94	36.15	35.62	42.52
	CommitMoE(w OWA)	52.24	51.77	39.41	36.17	35.69	43.06
DeepSeek-V2-Lite-Chat	Base	*69.60*	*46.02*	*37.58*	*49.12*	*36.72*	*47.81*
	CommitMoE(w/o OWA)	68.92	47.25	39.04	49.02	36.55	48.16
	CommitMoE(w Neutral OWA[2])	69.21	46.38	38.68	48.09	36.42	47.75
	CommitMoE(w OWA)	70.05	47.49	39.17	49.06	36.65	48.48

[1]R1 and R2 denote the Rouge-1 and Rouge-2 scores, respectively.
[2]Neutral OWA refers to the configuration where $\alpha_1 = \alpha_2 = 1.0$.

Table 3: Performance comparison of CommitMoE against original models and different OWA variants across different tasks. All values are represented as percentages

Mixtral-8x7B-Instruct. OWA is triggered when the number of correctly predicted experts falls within model-specific ranges: 1 for Mixtral-8x7B-Instruct, 2 to 3 for Qwen1.5-MoE-Chat, and 4 to 5 for DeepSeek-V2-Lite-Chat. Table 3 presents the performance of CommitMoE compared to the original models across our evaluation datasets.

The results in Table 3 demonstrate that CommitMoE effectively preserves model quality across different architectures while revealing model-specific patterns in OWA effectiveness. For Mixtral-8x7B-Instruct, the variant without OWA achieves the best average performance (51.02%), outperforming both the base model (50.85%) and the OWA variant (50.73%). This interesting phenomenon suggests that for some model architectures, the expert substitution strategy employed by CommitMoE may introduce a beneficial regularization effect. For DeepSeek-V2-Lite-Chat, the OWA variant shows the best average performance (48.48%), exceeding the base model (47.81%) across most metrics, notably achieving higher scores on both TruthfulQA metrics and GSM8K. For Qwen1.5-MoE-Chat, the OWA mechanism helps narrow the GSM8K performance gap from 2.88 to 0.98 percentage points compared to the base model, though its average performance (43.06%) remains slightly below the base model (43.17%). Further analysis of the Qwen1.5-MoE-Chat model shows that applying OWA increased the cosine similarity between outputs and those from the original router by up to 19% compared to the non-OWA variant, highlighting OWA's ability to preserve output consistency. These results validate our key insights regarding the dual nature of router behavior: predictable expert selection in high-certainty scenarios and output robustness to expert variations in low-certainty cases.

The model-specific OWA hyperparameters demonstrate significant improvements over the neutral configuration, where both α_1 and α_2 are set to 1.0. Across all models, manually configured values of α_1 and α_2 consistently outperform the baseline, underscoring the importance of selecting these parameters to match model-specific characteristics for improved downstream performance.

5.3 End-to-End Performance

We evaluated CommitMoE's end-to-end inference latency in two hardware environments using three representative MoE models (Mixtral-8x7B-Instruct, Qwen1.5-MoE-Chat, and DeepSeek-V2-Lite-Chat, denoted as Mixtral, Qwen, and DeepSeek, respectively, in Figure 5) across four diverse tasks (BBH, GSM8K, TruthfulQA, and WMT16-de-to-en). For BBH and GSM8K, we generated new 128 tokens, while for TruthfulQA and WMT16-de-to-en, we generated new 32 tokens. Following prior works (Eliseev and Mazur 2023; Hwang et al. 2024; Yi et al. 2025; Tang et al. 2024; Kong et al. 2024), we configure our experiments in a GPU-centric manner with a batch size of one, which aligns with typical usage patterns in resource-constrained environments such as edge-based continuous inference scenarios. To ensure the reliability and consistency of experimental results, we locked the frequencies of both CPU and GPU, eliminating performance fluctuations that might be caused by dynamic hardware frequency adjustments. To isolate the contributions of the prediction accuracy improvements versus our fallback-free strategy, we designed the "+Prefetch" variant as an ablation study, which replaces MoE-Infinity's prefetching component with Commit Router while still maintaining the fallback mechanism. The end-to-end inference speedup achieved by CommitMoE across different models and tasks is presented in Figure 5.

In Environment 1, CommitMoE consistently outperforms both vLLM and MoE-Infinity across all models and tasks. For Mixtral, CommitMoE achieves speedups of 1.3-1.5× over vLLM, with the highest gain observed on the WMT16-de-to-en dataset. For Qwen and DeepSeek models with more experts per layer, the speedups are even more substantial, reaching up to 1.8×. Notably, CommitMoE consistently outperforms the "+Prefetch" variant, highlighting the effective-

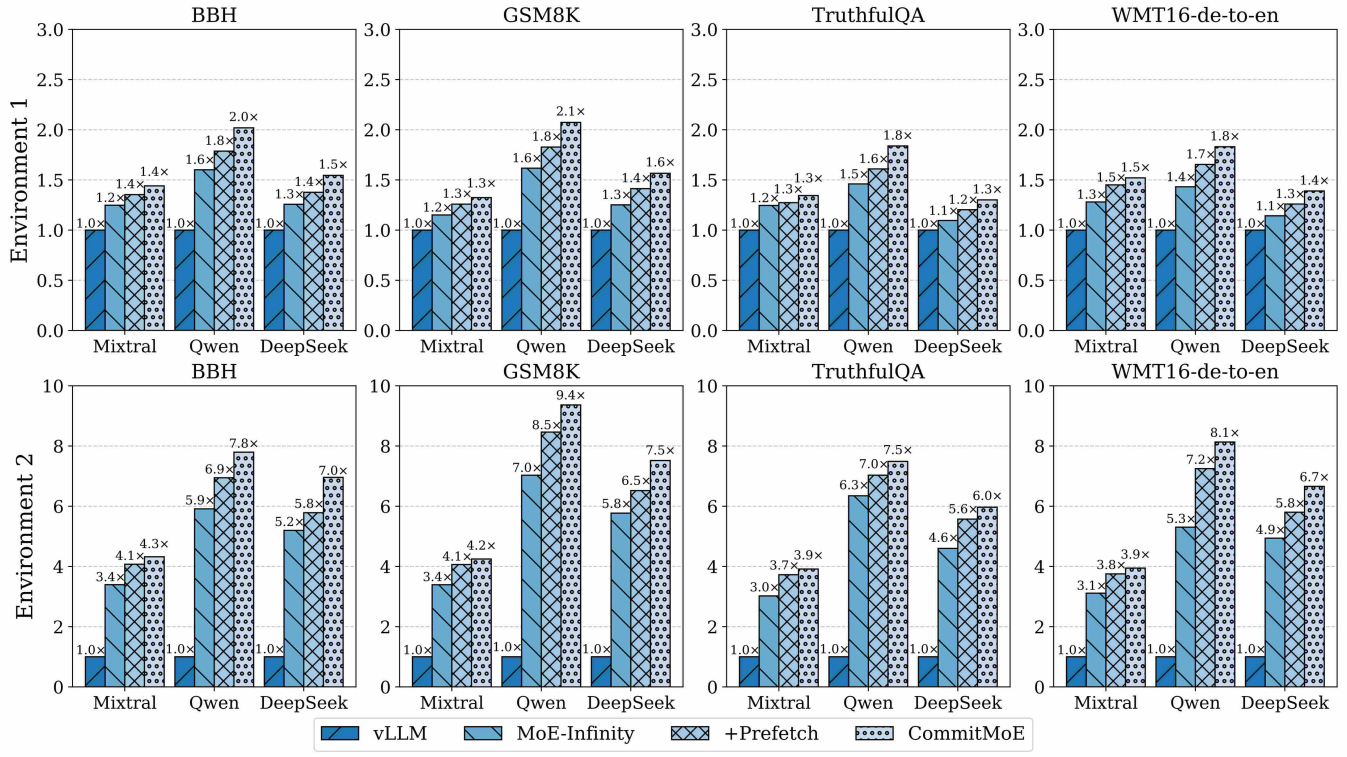

Figure 5: End-to-end inference speedup of CommitMoE compared to vLLM, MoE-Infinity, and +Prefetch across two different hardware environments.

ness of our fallback-free commitment strategy in eliminating expert loading delays.

The performance gap widens dramatically in the more constrained Environment 2. Here, CommitMoE achieves remarkable speedups of up to 4.3× for Mixtral, 9.4× for Qwen, and 7.5× for DeepSeek, compared to vLLM. This significant improvement in computing resource-constrained settings underscores CommitMoE's ability to effectively mitigate the PCIe bandwidth bottleneck through its commitment-based approach, making it particularly valuable for deployment scenarios with limited hardware resources.

To assess the statistical significance of our performance improvements, we conducted Wilcoxon signed-rank tests across all experimental configurations. The tests demonstrate significant improvements with p-values below 0.001 across all baseline comparisons, with effect sizes indicating practical significance through Cohen's d absolute values ranging from 0.567 to 7.356. These results confirm that the observed performance gains represent algorithmic advantages rather than statistical noise.

6 Discussion

The results in Table 3 reveal that OWA's effectiveness correlates with the number of activated experts per token. Models with higher expert activation counts, such as DeepSeek-V2-Lite-Chat with six activated experts, demonstrate greater benefits from the weight redistribution mechanism compared to models with lower activation counts like Mixtral-8x7B-Instruct with two activated experts. This observation suggests that OWA's value scales with the complexity of activated expert selection, where models with larger activation patterns can better leverage the score adjustment process. For practical deployment, we recommend enabling OWA for models with four or more activated experts per token, a common trend in recent architectures, where improvements are consistently observed. Both α_1 and α_2 are empirically tuned within a manageable range of 1.0 to 1.4 through grid search on a validation set, with optimal values varying across different model architectures.

7 Conclusions

In this paper, we introduced CommitMoE, an efficient MoE inference framework that avoids fallback mechanisms through a novel commitment-based strategy. CommitMoE exploits the insight that router certainty correlates with prediction accuracy, while model outputs remain robust to expert selection in low-certainty cases. This enables our Commit Router to make fallback-free decisions, eliminating unnecessary data transfer. The Output Weight Adjustment mechanism preserves model quality without retraining. Extensive evaluations demonstrate that CommitMoE achieves 1.3× to 9.4× faster inference compared to state-of-the-art offloading frameworks while maintaining model quality. We hope our work facilitates the deployment of MoE models in GPU-constrained environments.

Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work was sponsored by Doubao Fund.

References

Anthropic, A. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1: 1.

Bojar, O.; Chatterjee, R.; Federmann, C.; Graham, Y.; Haddow, B.; Huck, M.; Jimeno-Yepes, A.; Koehn, P.; Logacheva, V.; Monz, C.; Negri, M.; Névéol, A.; Neves, M.; Popel, M.; Post, M.; Rubino, R.; Scarton, C.; Specia, L.; Turchi, M.; Verspoor, K. M.; and Zampieri, M. 2016. Findings of the 2016 Conference on Machine Translation. In *Conference on Machine Translation*.

Cao, S.; Liu, S.; Griggs, T.; Schafhalter, P.; Liu, X.; Sheng, Y.; Gonzalez, J. E.; Zaharia, M.; and Stoica, I. 2025. Moe-lightning: High-throughput moe inference on memory-constrained gpus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 715–730.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Cui, W.; Han, Z.; Ouyang, L.; Wang, Y.; Zheng, N.; Ma, L.; Yang, Y.; Yang, F.; Xue, J.; Qiu, L.; et al. 2023. Optimizing dynamic neural networks with brainstorm. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, 797–815.

Dai, D.; Deng, C.; Zhao, C.; Xu, R. X.; Gao, H.; Chen, D.; Li, J.; Zeng, W.; Yu, X.; Wu, Y.; Xie, Z.; Li, Y. K.; Huang, P.; Luo, F.; Ruan, C.; Sui, Z.; and Liang, W. 2024. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. arXiv:2401.06066.

Eliseev, A.; and Mazur, D. 2023. Fast Inference of Mixture-of-Experts Language Models with Offloading. arXiv:2312.17238.

Frantar, E.; and Alistarh, D. 2023. QMoE: Practical Sub-1-Bit Compression of Trillion-Parameter Models. arXiv:2310.16795.

Gao, L.; Tow, J.; Abbasi, B.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; Le Noac'h, A.; Li, H.; McDonell, K.; Muennighoff, N.; Ociepa, C.; Phang, J.; Reynolds, L.; Schoelkopf, H.; Skowron, A.; Sutawika, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2024. The Language Model Evaluation Harness.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

He, S.; Dong, D.; Ding, L.; and Li, A. 2025. Towards Efficient Mixture of Experts: A Holistic Study of Compression Techniques. arXiv:2406.02500.

Hwang, R.; Wei, J.; Cao, S.; Hwang, C.; Tang, X.; Cao, T.; and Yang, M. 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)*, 1018–1031. IEEE.

Imani, H.; Amirany, A.; and El-Ghazawi, T. 2024. Mixture of Experts with Mixture of Precisions for Tuning Quality of Service. arXiv:2407.14417.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Hanna, E. B.; Bressand, F.; Lengyel, G.; Bour, G.; Lample, G.; Lavaud, L. R.; Saulnier, L.; Lachaux, M.-A.; Stock, P.; Subramanian, S.; Yang, S.; Antoniak, S.; Scao, T. L.; Gervet, T.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2024. Mixtral of Experts. arXiv:2401.04088.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kong, R.; Li, Y.; Feng, Q.; Wang, W.; Ye, X.; Ouyang, Y.; Kong, L.; and Liu, Y. 2024. SwapMoE: Serving Off-the-shelf MoE-based Large Language Models with Tunable Memory Budget. arXiv:2308.15030.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180.

Lin, S.; Hilton, J.; and Evans, O. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. arXiv:2109.07958.

Liu, A.; Feng, B.; Wang, B.; Wang, B.; Liu, B.; Zhao, C.; Dengr, C.; Ruan, C.; Dai, D.; Guo, D.; et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Lu, M.; Sun, J.; Lin, J.; Zhou, Z.; and Sun, G. 2025. LuaLLM: Learning Unstructured-Sparsity Allocation for Large Language Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lu, X.; Liu, Q.; Xu, Y.; Zhou, A.; Huang, S.; Zhang, B.; Yan, J.; and Li, H. 2024. Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models. arXiv:2402.14800.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703.

Rajbhandari, S.; Li, C.; Yao, Z.; Zhang, M.; Aminabadi, R. Y.; Awan, A. A.; Rasley, J.; and He, Y. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, 18332–18346. PMLR.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. arXiv:1701.06538.

Sheng, Y.; Zheng, L.; Yuan, B.; Li, Z.; Ryabinin, M.; Fu, D. Y.; Xie, Z.; Chen, B.; Barrett, C.; Gonzalez, J. E.; Liang, P.; Ré, C.; Stoica, I.; and Zhang, C. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. arXiv:2303.06865.

Song, X.; Zhong, Z.; Chen, R.; and Chen, H. 2025. ProMoE: Fast MoE-based LLM Serving using Proactive Caching. arXiv:2410.22134.

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A. M.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. 2022. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q. V.; Chi, E. H.; Zhou, D.; ; and Wei, J. 2022. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. *arXiv preprint arXiv:2210.09261*.

Tang, P.; Liu, J.; Hou, X.; Pu, Y.; Wang, J.; Heng, P.-A.; Li, C.; and Guo, M. 2024. HOBBIT: A Mixed Precision Expert Offloading System for Fast MoE Inference. arXiv:2411.01433.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: An instruction-following llama model.

Team, G.; Georgiev, P.; Lei, V. I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Wang, S.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Team, Q. 2024. Qwen1.5-MoE: Matching 7B Model Performance with 1/3 Activated Parameters.

Wei, Y.; Du, J.; Jiang, J.; Shi, X.; Zhang, X.; Huang, D.; Xiao, N.; and Lu, Y. 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*, 1–14. IEEE.

Xie, Y.; Zhang, Z.; Zhou, D.; Xie, C.; Song, Z.; Liu, X.; Wang, Y.; Lin, X.; and Xu, A. 2024. MoE-Pruner: Pruning Mixture-of-Experts Large Language Model using the Hints from Its Router. arXiv:2410.12013.

Xue, L.; Fu, Y.; Lu, Z.; Mai, L.; and Marina, M. 2025. MoE-Infinity: Efficient MoE Inference on Personal Machines with Sparsity-Aware Expert Cache. arXiv:2401.14361.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; and Fan, Z. 2024. Qwen2 Technical Report. arXiv:2407.10671.

Yi, R.; Guo, L.; Wei, S.; Zhou, A.; Wang, S.; and Xu, M. 2025. EdgeMoE: Empowering Sparse Large Language Models on Mobile Devices. arXiv:2308.14352.

Yuan, Y.; Ma, L.; and Talati, N. 2025. MoE-Lens: Towards the Hardware Limit of High-Throughput MoE LLM Serving Under Resource Constraints. arXiv:2504.09345.

Zhong, S.; Liang, L.; Wang, Y.; Wang, R.; Huang, R.; and Li, M. 2024. 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 '24, 1–9. ACM.