EC²MoE: Adaptive End-Cloud Pipeline Collaboration Enabling Scalable Mixture-of-Experts Inference

Zheming Yang¹, Yunqing Hu^{1,3}, Sheng Sun¹, Wen Ji^{1,2}

¹Institute of Computing Technology, Chinese Academy of Sciences

²Institute of Al for Industries

³University of Chinese Academy of Sciences

Abstract

The Mixture-of-Experts (MoE) paradigm has emerged as a promising solution to scale up model capacity while maintaining inference efficiency. However, deploying MoE models across heterogeneous end-cloud environments poses new challenges in expert scheduling, communication overhead, and resource heterogeneity. In this paper, we propose EC^2MoE , an adaptive framework for scalable MoE inference via end-cloud pipeline collaboration. First, we design a hardware-aware lightweight group gate network that enhances expert selection and computational efficiency. By incorporating a hardware-aware local expert selection mechanism, the system adaptively filters candidate experts based on real-time device profiles. A lightweight group gate module then integrates local and global gating outputs to achieve high-quality expert routing with minimal overhead. Second, we develop a pipeline optimization mechanism based on endcloud collaboration to accelerate MoE inference. This includes an encoder-decoder structure based on low-rank compression, which reduces transmission and computation costs. And a route-aware heuristic pipeline scheduling algorithm that dynamically allocates inference stages across devices according to workload and network topology. Extensive experiments show that EC^2MoE can increase throughput by $2.2 \times$ to $5.1 \times$ and reduce end-to-end latency by 53% to 67% while maintaining high accuracy compared to state-of-the-art methods. It also maintains good scalability under dynamic load and network environments.

Introduction

In recent years, the demand for large-scale deep learning models has grown dramatically (Ge et al. 2023; Shen et al. 2024a; Menghani 2023), driven by the rapid advancement of AI applications in areas such as natural language understanding, computer vision, and multi-modal reasoning (Liang et al. 2024). To meet this demand, researchers have explored various model scaling strategies (Hwang et al. 2023; Chen et al. 2024; Yin et al. 2024). Among them, the Mixture-of-Experts (MoE) architecture has emerged as a particularly promising solution (Zhou et al. 2022). By selectively activating a sparse subset of expert networks during inference, MoE models enable substantial increases in parameter count—often reaching hundreds of billions—without incurring a proportional increase in computational cost (Chen et al. 2022; Riquelme et al. 2021). This

sparsity-aware computation makes MoE architectures well-suited for balancing inference efficiency and model expressiveness (Szatkowski et al. 2024; Liu et al. 2025), enabling state-of-the-art performance across a range of complex AI tasks.

Despite these advantages, efficiently deploying MoE models in real environments remains challenging (Liu, Wang, and Wu 2025). Unlike conventional single models that can be easily compressed or quantized for end deployment, MoE models consist of multiple dynamically invoked expert models, whose activation patterns vary per input and require efficient gating and routing mechanisms (Raibhandari et al. 2022). This variability introduces new difficulties in system-level resource scheduling, model placement, and expert communication (Gale et al. 2023; Cao et al. 2025). On one hand, resource-constrained end devices struggle to support the intensive computational demands of high-capacity MoE backbones, especially when multiple expert paths are involved (Shen et al. 2024b; Jin et al. 2025). This results in degraded inference accuracy or throughput if only local computation is used (Li et al. 2025). On the other hand, relying solely on the cloud introduces latency overhead and may lead to underutilization of local resources (Deshpande et al. 2024). Moreover, cloud-only execution becomes highly sensitive to network fluctuations (Yu et al. 2024), making it difficult to guarantee consistent performance in latency-critical applications. These issues are further exacerbated by fluctuating network conditions, device heterogeneity, and dynamic workload patterns commonly encountered in real-world sce-

To address these challenges, we propose EC^2MoE , a novel adaptive framework that enables scalable and efficient MoE inference through end-cloud pipeline collaboration in heterogeneous distributed environments. EC^2MoE integrates hardware-aware expert selection with coordinated execution across end and cloud, effectively mitigating resource constraints, network variability, and communication overhead. By jointly optimizing expert routing and inference scheduling, our framework achieves high throughput, low latency, and robust scalability under dynamic workloads. Our main contributions are as follows:

 We propose a hardware-aware lightweight group gate network that efficiently adapts MoE expert selection to heterogeneous end-cloud environments. By incorporating a local expert selection mechanism based on device hardware characteristics and designing a lightweight group gate network module, it significantly reduces inference latency and routing overhead while maintaining expert selection quality.

- We develop a collaborative end-cloud pipeline optimization mechanism tailored for scalable MoE inference.
 It integrates a low-rank compression-based encoder-decoder to reduce transmission costs and a route-aware heuristic pipeline scheduler that dynamically maps inference sub-tasks across end and cloud based on work-load and communication patterns, maximizing overall throughput.
- We evaluate the performance of the framework and compare it with mainstream baseline methods. Experimental results show that EC^2MoE can increase throughput by $2.2 \times$ to $5.1 \times$ and reduce end-to-end latency by 53% to 67%, and without sacrificing accuracy.

Related Works

Cloud-based MoE inference optimization

Cloud-based MoE inference optimization has been widely studied due to the abundant computing resources and scalability of cloud infrastructures (Hwang et al. 2024; Hu et al. 2025). Early works primarily focused on efficient expert routing and load balancing to minimize the communication and computation overhead during inference. For example, GShard (Lepikhin et al. 2020) and Switch Transformer (Fedus, Zoph, and Shazeer 2022) introduced sparse expert activation and simplified gating mechanisms to enable large-scale MoE training and inference in cloud environments. These methods significantly improved model scalability while controlling inference costs by activating only a subset of experts per input. Subsequent research further explored expert placement and communication optimization. Tutel (Hwang et al. 2023) and EfficientMoE (Zeng et al. 2025) implemented expert parallelism strategies and expert sharding to minimize cross-device communication during MoE inference, enhancing throughput and reducing latency. In addition, model parallelism frameworks such as DeepSpeedMoE (Dai et al. 2024) and Fsmoe (Pan et al. 2025) provided system-level optimizations for cloud-based deployment by improving the scheduling of expert computation and network transfer. These works, however, often assume homogeneity and high bandwidth availability in cloud clusters, which may not generalize well to hybrid or end scenarios.

End-based MoE inference optimization

. Deploying MoE models on resource-constrained end devices faces limitations in terms of memory and computing power. To address this, previous studies have proposed various lightweight and dynamic management methods to improve inference efficiency. EdgeMoE (Yi et al. 2025) and D²MoE (Wang et al. 2025) achieve expert selection tailored to input and device status through sparse activation and dynamic routing strategies. Edge-MoE (Sarkar et al. 2023), eMoE (Tairin et al. 2025), and Fate (Fang et al. 2025) focus

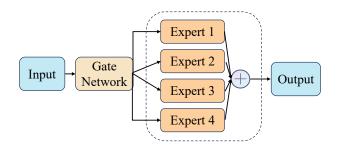


Figure 1: The illustration of MoE architecture, where a gating network dynamically selects a subset of expert networks to process each input. The selected expert outputs are then aggregated to form the final prediction.

on memory optimization, employing mechanisms such as expert sharing and cross-layer gating to reduce model overhead. AdapMoE (Zhong et al. 2024) introduces a sensitivity-based gating mechanism to enable flexible precision control during inference. To address runtime resource fluctuations, the work (Kong et al. 2025) proposes a dynamic expert replacement mechanism, while Flame (Lin et al. 2024) fully leverages MoE sparsity on FPGAs to achieve efficient hardware deployment. Although these methods significantly enhance end-side MoE inference capabilities, they remain constrained by model capacity and flexibility, making it challenging to meet the demands of high-precision tasks.

Preliminary

Mixture-of-Experts

MoE is an architecture that enhances model capacity and efficiency by dynamically activating submodels (experts), as shown in Figure 1. Its core idea is to route input samples to a small number of relevant expert networks (such as feedforward layers) for processing, rather than having all parameters participate in the computation. By using a learnable gating mechanism (Gating Network) to select the top-k experts, MoE significantly increases the number of parameters while keeping the computational load close to that of a dense model. The core of MoE lies in dynamically activating expert submodels through a gating network. Given an input x, its output y can be represented as:

$$y = \sum_{i=1}^{n} G(x)_i \cdot E_i(x) \tag{1}$$

where E_i denotes the i nd expert network (typically an independent feedforward layer), and G(x) is the gating function output weight vector, satisfying $\sum_i G(x)_i = 1$. A key advantage of MoE is its ability to perform effective pretraining with far fewer computational resources than dense models require. Compared to dense models, hybrid expert models typically achieve the same quality level faster. However, they may face challenges related to expert load balancing and cross-device communication overhead, especially in distributed deployments.

End-Cloud Architecture

The end-cloud architecture represents a distributed computing paradigm that strategically allocates computational workloads between resource-constrained end devices (e.g., smartphones and IoT sensors) and centralized cloud servers through collaborative execution mechanisms. In this architecture, end devices primarily perform latency-sensitive operations (e.g., real-time data preprocessing and lightweight model inference), while computationally intensive tasks (e.g., large-scale model training and complex inference) are offloaded to cloud servers. This hierarchical computation approach effectively addresses critical challenges in bandwidth utilization and end-to-end latency optimization. Particularly for MoE model deployment, the end-cloud architecture provides essential infrastructure support, as MoE systems inherently require distributed processing capabilities to handle their characteristic large-scale, diverse task distributions and dynamic expert routing mechanisms.

Method

EC2MoE Overall Design

The overall workflow of EC^2MoE consists of two key components: (1) Hardware-Aware Lightweight Group Gate Network (HL-GGN) and (2) Pipeline Optimization based on End-Cloud Collaboration (PO-ECC). As shown in Figure 2, first, the system performs hardware-aware local expert selection on the end, where candidate experts are filtered based on the device's real-time resource profile to reduce unnecessary computation and transmission. Then, a lightweight group gate network fuses local gating signals with global expert routing to generate high-quality expert selection decisions with minimal overhead. Second, the selected inputs and routing information are passed through an end-cloud collaborative pipeline, where an encoderdecoder module based on low-rank compression reduces transmission latency and bandwidth cost, while a routeaware pipeline scheduler dynamically distributes inference stages across edge and cloud devices. This end-to-end collaborative workflow ensures optimal utilization of heterogeneous resources, improves system throughput, and maintains low latency under dynamic workloads and network conditions.

Hardware-Aware Lightweight Group Gate Network

To enable efficient expert selection in resource-constrained end environments, we propose the **HL-GGN**, which consists of two key components: (1) Hardware-Aware Local Expert Selection and (2) Lightweight Group Gate Network. This design ensures that only the most relevant experts are activated while minimizing computational overhead.

Local Expert Selection In heterogeneous end-cloud collaborative inference scenarios, end devices have limited computing power and cannot evaluate all expert models simultaneously during each inference. Therefore, we introduce a hardware-aware local expert selection mechanism to dynamically narrow down the set of expert candidates while

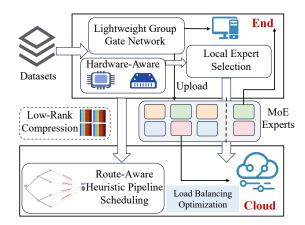


Figure 2: The overall design of the proposed EC^2MoE framework.

maintaining the flexibility of the MoE architecture. Specifically, we define a hardware-aware function $H(\cdot)$, whose input is the real-time state vector of the device:

$$S_{\rm device} = \{C_{cpu}, M_{\rm mem}, P_{\rm power}, B_{\rm bandwidth}\} \qquad (2)$$
 where C_{cpu} denotes the currently available CPU resources, M_{mem} represents the memory status, P_{power} indicates the current battery level or power consumption limit, and $B_{bandwidth}$ signifies the network bandwidth condition. Based on the above states, the hardware-aware function can predict the current device's inference capability threshold $T_{capability}$:

$$T_{\text{capability}} = H\left(S_{\text{device}}\right)$$
 (3)

Subsequently, the computational complexity characteristics of the expert subnetwork are defined as vector V_{expert} , and by comparing the complexity characteristics of each expert network with the current device capability threshold, a subset of experts \mathcal{E}_{local} that meet the local execution conditions is selected:

$$\mathcal{E}_{local} = \left\{ e_i \mid f\left(V_{expert_i}, T_{capability}\right) \le \epsilon, \forall e_i \in \mathcal{E} \right\} \quad (4)$$

Here, $f(\cdot)$ is the complexity matching function, and ϵ is the set complexity tolerance threshold. Through this hardware-aware selection mechanism, we ensure that end devices only execute inference tasks within their capability range, while other high-complexity tasks are offloaded to the cloud, effectively balancing the real-time performance and resource consumption of inference.

Lightweight Group Gate Network Traditional gate networks generally calculate the selection probability of each expert network through a single fully connected layer (Ma et al. 2018). However, this design has the following shortcomings in end devices: (1) The weight matrix W_g is linearly related to the number of experts and feature dimensions, making it difficult to deploy effectively in resource-constrained end devices (Petersen et al. 2022). (2) High-dimensional matrix multiplication significantly increases inference latency.

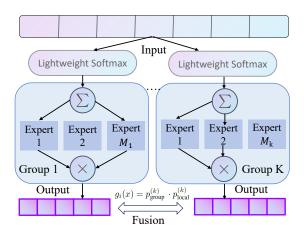


Figure 3: The overview of the lightweight group gate network.

To address the above issues, we first employ a group gate mechanism to divide the expert network into several groups, with each group sharing a separate gate subnetwork, thereby significantly reducing the number of parameters and computational cost of the gate network, as shown in Figure 3. Specifically, the M experts are divided into K groups (each group contains M_k experts, where $M = \sum_{k=1}^K M_k$), and each group independently learns a lightweight Softmax gate network:

$$g^{(k)}(x) = \operatorname{Softmax}(W_k x + b_k),$$

$$W_k \in \mathbb{R}^{M_k \times d}, b_k \in \mathbb{R}^{M_k}$$
(5)

Since the number of experts M_k corresponding to each gate-controlled subnetwork is significantly smaller than the total number of experts M, the number of parameters and computational load for each subnetwork are greatly reduced. Furthermore, to achieve more flexible and accurate expert selection, we introduce a two-stage dynamic fusion strategy. In stage 1, a low-dimensional global Softmax gated network is used to quickly determine the overall contribution of each group of experts:

$$p_{\text{group}} = \operatorname{Softmax} (W_{\text{global}} x + b_{\text{global}}),$$

$$W_{\text{global}} \in \mathbb{R}^{K \times d}, b_{\text{global}} \in \mathbb{R}^{K}$$
(6)

Then, in Stage 2, within each expert group, the aforementioned lightweight grouped gating subnetwork is used to calculate the group-internal expert probability $p_{local}^{(k)}$. Finally, by multiplying and fusing the group-level and group-internal probabilities through the two-stage gating probabilities, the final expert selection probability is obtained:

$$g_i(x) = p_{\text{group}}^{(k)} \cdot p_{\text{local},i}^{(k)}, \quad i \in \text{group } k$$
 (7)

Through this two-stage mechanism, it can dynamically focus on a small number of experts within highly correlated expert groups, further improving inference efficiency while maintaining good expert selection accuracy.

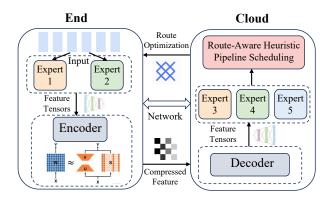


Figure 4: The overview of the pipeline optimization based on end-cloud collaboration.

Pipeline Optimization based on End-Cloud Collaboration

To achieve efficient and scalable MoE inference in an endcloud collaborative environment, we propose the **PO-ECC** that combines low-rank compression for encoder-decoder and route-aware pipeline scheduling, as shown in Figure 4. It can ensure minimal communication overhead while maximizing computational parallelism.

Encoder-Decoder based on Low-Rank CompressionDuring the inference process involving frequent interactions

During the inference process involving frequent interactions between the end and the cloud, the transmission of a large number of intermediate feature tensors causes severe bandwidth pressure and communication latency (Hou et al. 2015; Idelbayev and Carreira-Perpinán 2020). To address this issue, we designed an encoder-decoder model based on low-rank compression to compress and reconstruct intermediate features, thereby reducing the communication burden between the end and the cloud.

Specifically, let the input feature tensor be $X \in \mathbb{R}^{h \times w \times c}$. We construct a lightweight compression module at the end to project it into a low-rank subspace, yielding the compressed representation $Z = U^{\top}XV$, where $U \in \mathbb{R}^{h \times r}$, $V \in \mathbb{R}^{w \times r}$ is an adaptively optimized low-rank projection matrix, and $r \ll \min(h, w)$ denotes the compression dimension.

On the cloud, the decoder module uses the corresponding inverse projection matrices \hat{U}, \hat{V} to reconstruct the original feature tensor $\hat{X} = \hat{U}Z\hat{V}^{\top}$. To mitigate the accuracy degradation caused by reconstruction errors, we adopt an end-cloud joint training approach to minimize the following objective function:

$$\mathcal{L}_{\text{rec}} = \|X - \hat{X}\|_2^2 + \lambda \cdot \mathcal{L}_{\text{task}}(\hat{X})$$
 (8)

where \mathcal{L}_{task} is the task loss function, and λ controls the trade-off between compression and task accuracy. This method effectively compresses communication data while maintaining accuracy, significantly reduces transmission latency between the end and cloud, and provides a solid foundation for subsequent pipeline scheduling.

Route-Aware Heuristic Pipeline Scheduling To further improve the overall processing efficiency of pipeline tasks in an end-to-end cloud collaboration environment, we propose a route-aware heuristic pipeline scheduling algorithm. Specifically, the pipeline scheduling problem can be abstracted as follows: given a set of tasks $\mathcal{T}=t_1,t_2,...,t_N$, each task can be assigned to either end or cloud processing, and different tasks have different computational complexities $C(t_i)$ and communication costs $Comm(t_i)$. The heuristic optimization objective is defined as:

$$\min \sum_{i=1}^{N} \left[\alpha \cdot \text{ExecTime} \left(t_i \right) + (1 - \alpha) \cdot \text{Comm} \left(t_i \right) \right]$$
 (9)

where $ExecTime(t_i)$ denotes the expected execution time of task t_i when processed on the end or in the cloud. $Comm(t_i)$ represents the communication cost of task transmission. The parameter α is used to adjust the trade-off between computation and communication. To effectively solve the above optimization problem, we use a heuristic algorithm based on greedy selection for pipeline scheduling. First, we calculate the priority $P(t_i)$ of each task:

$$P(t_i) = \frac{C(t_i)}{\text{Comm}(t_i) + \epsilon}$$
 (10)

where ϵ is a small constant to prevent division by zero. Tasks with higher priorities tend to be executed locally to reduce communication costs.

Then, based on the current end device load $Load_{end}$, cloud load $Load_{cloud}$, and the priority $P(t_i)$ of each task, the heuristic algorithm decides the execution location of the task:

Location
$$(t_i) = \begin{cases} \text{End,} & \text{if Load }_{\text{end}} + C(t_i) \leq T_{\text{end}} \\ & \text{and } P(t_i) \geq \beta \\ \text{Cloud,} & \text{otherwise} \end{cases}$$
(11)

where T_{end} is the maximum tolerable computation threshold for end devices, and β is the priority threshold for local task execution. This heuristic rule enables rapid task allocation decisions and reduces task waiting latencys. Through the above routing-aware heuristic pipeline scheduling algorithm, it can efficiently respond to the dynamic nature of heterogeneous communication environments and improve the overall inference efficiency of the end-to-cloud pipeline.

Performance Evaluation

Evaluation Setup

Datasets and Implementation Details To evaluate the effectiveness of the proposed framework, we conduct experiments on two widely used benchmark datasets: the GLUE (Wang et al. 2019) and the SQuAD (Rajpurkar et al. 2016). We adopt the Switch Transformer as our base MoE model. The MoE inference pipeline is implemented using PyTorch, with custom modules for expert partitioning, dynamic routing, and end-cloud collaboration. Hardware simulations include an Intel Xeon Silver 4214R CPU for end devices and

two NVIDIA A100 GPUs for cloud servers, reflecting a realistic heterogeneous compute environment. To simulate real network conditions, we use Linux Traffic Control to introduce network bandwidth settings. The communication between end and cloud is set under a 300 Mbps network with a 20% fluctuation to reflect dynamic bandwidth conditions.

Baseline Methods. We compare our solution with the following four baseline methods. (1) **BrownoutServe:** This is a cloud-based MoE inference service framework that reduces the number of expert visits and lowers inference latency by introducing a joint expert mechanism. (2) **EdgeMoE:** This is an MoE inference engine based on end devices that achieves memory and computational efficiency by dividing the model into a storage hierarchy.

Evaluation Metrics: Three key metrics are employed to comprehensively assess the performance of the proposed framework: accuracy, end-to-end latency, and throughput. Accuracy is used to evaluate the correctness of model predictions and ensure that efficiency gains do not compromise model performance. End-to-end latency measures the total time from input reception at the end to final output generation, capturing the real-time responsiveness of the end-cloud collaborative inference pipeline. Throughput quantifies the number of inference tasks completed per unit time, reflecting the system's overall processing efficiency.

Parameter Selection: The number of experts in the MoE models is set to 8, 16, 32, and 64, enabling the assessment of the system's scalability and its adaptability to models of increasing complexity. The input sequence length is fixed at 256 tokens, which aligns with common benchmarks in language modeling tasks and ensures a consistent inference context across all experiments. A uniform batch size of 4 is used to balance computational efficiency and memory usage, especially under resource-constrained end settings. To reduce routing complexity and maintain inference efficiency, the gating mechanism adopts a Top-1 expert selection policy, where only the expert with the highest activation score is selected for each input, thereby minimizing redundant computation. Furthermore, to reflect hardware-aware constraints on the end side, a local expert selection mechanism is employed, with a selection cap that restricts the candidate expert set to at most 40% of the total experts. This constraint ensures that only a small subset of experts is evaluated on the end device, significantly reducing computation and communication overhead while retaining the flexibility and performance benefits of sparse expert activation.

End-to-End Results

Accuracy Comparison The experimental results are shown in Table 1. The EC^2MoE method proposed in this paper achieves the highest accuracy on both datasets. Compared with EdgeMoE, the average improvement reaches 4.07% and 4.1%. More importantly, the proposed method not only outperforms EdgeMoE but also surpasses the purely cloud-based method BrownoutServe. This is because cloud-based methods often assume stable network transmission and rely on global expert activation. However, in real-world end-cloud environments, high latency and bandwidth fluctuations may limit expert selection for certain inputs,

Table 1: The accuracy comparison results under the different datasets.

Switch-Base	GLUE			SQuAD		
	BrownoutServe	EdgeMoE	EC^2MoE	BrownoutServe	EdgeMoE	EC^2MoE
8-experts	81.2	77.6	81.6	82.2	78.4	82.5
16-experts	81.7	77.9	82.3	82.1	77.9	82.3
32-experts	80.3	76.2	80.5	82.7	78.1	83.1
64-experts	80.8	77.8	81.4	82.3	77.3	82.6

thereby affecting inference integrity and final accuracy. This paper's method employs end-cloud joint gating and pipeline-level scheduling optimization to perform local selection and feature compression as inputs flow through the end, transmitting more representative information to the cloud and thereby reducing accuracy losses caused by network uncertainty. In contrast, EdgeMoE methods generally have lower accuracy rates due to the limited computing power and storage of end devices, which prevent the evaluation of the entire expert set, resulting in a limited number of activated experts and significant bias in the selection results. Although Edge-MoE avoids network transmission overhead locally, its disadvantages in expert diversity and model capacity utilization make it difficult to maintain the same inference accuracy as cloud-based methods in complex tasks.

Throughput Comparison The experimental results show that EC^2MoE demonstrates significant throughput advantages across all expert scales, as shown in Figure 5. Specifically, compared to BrownoutServe, EC2MoE achieves an average throughput improvement of over 2.2× at expert settings of 8, 16, 32, and 64. Compared to EdgeMoE, the average improvement reaches over 5.1×. The fundamental reason for this improvement lies in this paper introduces a routing-aware end-cloud asynchronous scheduling mechanism that fully leverages the heterogeneous parallel capabilities of the end and cloud to maximize inference process overlap and throughput. And hardware-aware expert selection strategies are employed to asynchronously transmit only critical information to the cloud, thereby significantly reducing communication overhead and redundant computational load. In contrast, BrownoutServe processes most of the inference process in the cloud, which, despite its strong computing power, is limited by network latency and bandwidth fluctuations and cannot efficiently process a large number of requests in parallel. EdgeMoE, on the other hand, relies entirely on the local computing power of terminal devices, which have a short inference path but limited processing capacity, especially when the number of experts increases, leading to a significant bottleneck in resources and a rapid decline in throughput.

Latency Comparison As shown in Figure 6, the experimental results show that our proposed method also has significant advantages in terms of latency performance. Compared to the BrownoutServe method, EC^2MoE can reduce the average latency by 67%. Compared to the EdgeMoE method, the average latency is reduced by 53%. EC^2MoE can be achieved through a routing-aware task scheduling

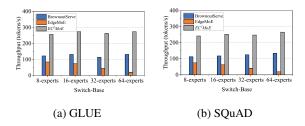


Figure 5: The throughput comparison of different methods under different numbers of experts.

mechanism, so that the end-side and cloud-side can work in parallel, thereby significantly reducing overall transmission and inference latency. Specifically, the high latency of the BrownoutServe method mainly stems from its high dependence on network transmission. In actual environments, network instability and latency fluctuations can easily cause transmission bottlenecks, thereby delaying the overall inference process. Especially in scenarios where the number of experts increases, the cost of cloud resource scheduling rises significantly, further increasing the system response time. While the EdgeMoE method avoids network transmission latency, it is constrained by the computational power of terminal devices, resulting in limited processing capabilities during expert selection and model execution. Especially when dealing with high-capacity MoE structures, computational bottlenecks can easily form.

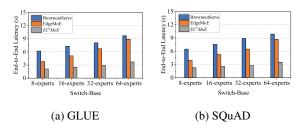


Figure 6: The latency comparison of different methods under different numbers of experts.

Scalable Analysis

Task Load Changes To assess the scalability of EC^2MoE under different reasoning load intensities, we simulated five request rates (Request Rate = 2, 4, 6, 8, 10 req/s) and sta-

tistically analyzed the average performance of throughput and end-to-end latency under different expert quantity settings, as shown in Figure 7. It can be seen that as the request rate gradually increases, the method proposed in this paper demonstrates superior linear scalability in throughput, maintaining steady growth even under high loads. At the same time, the increase in latency is significantly lower than that of the comparison methods. In contrast, Brownout-Serve is constrained by cloud processing resources and network congestion at high request rates, resulting in saturated throughput growth and a sharp increase in latency. Edge-MoE, on the other hand, quickly reaches a processing bottleneck in high-concurrency scenarios due to the limited computing power of end devices. The end-to-cloud pipeline collaboration mechanism adopted in this paper achieves high matching between processing capacity and input intensity by reasonably splitting the inference path and dynamically distributing task loads.

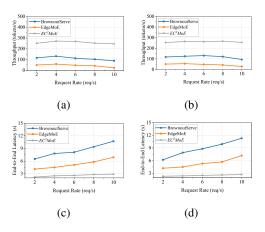


Figure 7: The test results under different request rates. (a) and (b) are throughput results under the GLUE and SQuAD datasets. (c) and (d) are latency results under the GLUE and SQuAD datasets.

Dynamic Network Environment To further validate the scalability of EC^2MoE in dynamic network environments, we set five network bandwidth fluctuation ranges (Bandwidth Fluctuation = 0%, 10%, 20%, 30%, 40%). The experimental results are shown in Figure 8. Even under conditions of increased network fluctuations, EC2MoE maintains stable throughput and latency, significantly outperforming BrownoutServe and EdgeMoE. BrownoutServe is highly sensitive to bandwidth fluctuations, and its cloudbased full-path dependency leads to significant latency increases and throughput collapse in high-jitter environments. While EdgeMoE does not rely on the network, its local inference capabilities cannot handle complex inference tasks with a high number of experts, resulting in limited throughput. In contrast, EC^2MoE effectively reduces dependence on bandwidth stability through its strategy of local expert selection and asynchronous transmission, demonstrating excellent scalability and robustness even under uncertain network conditions.

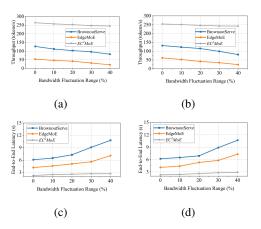


Figure 8: The test results under different bandwidth fluctuation. (a) and (b) are throughput results under the GLUE and SQuAD datasets. (c) and (d) are latency results under the GLUE and SQuAD datasets.

Ablation Studies

To validate the contribution of each core module to overall performance, we removed two key design components: (1) HL-GGN and (2) PO-ECC. While keeping all other settings unchanged, the system's accuracy, throughput, and latency changes were evaluated on the GLUE and SQuAD datasets. Experimental results show that after removing HL-GGN, the system could not adequately adapt to the resource state of the terminal device during the expert selection phase, leading to a decrease in expert activation accuracy, an average reduction in overall accuracy of 2.1%, and an increase in latency of approximately 23%. Removing PO-ECC prevents the inference path from asynchronously overlapping, causing communication and computation between the end and cloud to become blocked, resulting in an average throughput decrease of 38% and a 45% increase in end-to-end latency. These results indicate that both designs play a critical role in supporting system performance, collectively driving comprehensive improvements in performance and scalability of MoE inference.

Conclusion

In this paper, we propose EC^2MoE , the first framework to enable adaptive MoE inference via end-cloud pipeline collaboration. This work introduces a novel system architecture that jointly considers expert scheduling, communication efficiency, and hardware heterogeneity. Specifically, we design a hardware-aware lightweight group gate network for efficient and accurate expert routing in end-cloud systems. And we then develop a pipeline optimization mechanism that coordinates inference execution across end and cloud through low-rank compression and route-aware heuristic scheduling. Extensive experiments have shown that EC^2MoE significantly improves throughput and accuracy while reducing end-to-end latency. At the same time, it also maintains competitive scalability under dynamic workloads and network conditions.

References

- 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.
- Chen, L.; Davis, J. Q.; Hanin, B.; Bailis, P.; Stoica, I.; Zaharia, M. A.; and Zou, J. Y. 2024. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37: 45767–45790.
- Chen, Z.; Deng, Y.; Wu, Y.; Gu, Q.; and Li, Y. 2022. Towards understanding the mixture-of-experts layer in deep learning. *Advances in neural information processing systems*, 35: 23049–23062.
- Dai, D.; Deng, C.; Zhao, C.; Xu, R.; Gao, H.; Chen, D.; Li, J.; Zeng, W.; Yu, X.; Wu, Y.; et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.
- Deshpande, U.; Janssen, T.; Srivatsa, M.; and Sundararaman, S. 2024. MoEsaic: Shared Mixture of Experts. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, 434–442.
- Fang, Z.; Hong, Z.; Huang, Y.; Lyu, Y.; Chen, W.; Yu, Y.; Yu, F.; and Zheng, Z. 2025. Fate: Fast Edge Inference of Mixture-of-Experts Models via Cross-Layer Gate. *arXiv* preprint arXiv:2502.12224.
- Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120): 1–39.
- Gale, T.; Narayanan, D.; Young, C.; and Zaharia, M. 2023. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5: 288–304.
- Ge, Y.; Hua, W.; Mei, K.; Tan, J.; Xu, S.; Li, Z.; Zhang, Y.; et al. 2023. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36: 5539–5568.
- Hou, J.; Chau, L.-P.; Magnenat-Thalmann, N.; and He, Y. 2015. Sparse low-rank matrix approximation for data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(5): 1043–1054.
- Hu, J.; Xu, M.; Ye, K.; and Xu, C. 2025. BrownoutServe: SLO-Aware Inference Serving under Bursty Workloads for MoE-based LLMs. *arXiv preprint arXiv:2507.17133*.
- Hwang, C.; Cui, W.; Xiong, Y.; Yang, Z.; Liu, Z.; Hu, H.; Wang, Z.; Salas, R.; Jose, J.; Ram, P.; et al. 2023. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5: 269–287.
- 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.

- Idelbayev, Y.; and Carreira-Perpinán, M. A. 2020. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8049–8059.
- Jin, L.; Zhang, Y.; Li, Y.; Wang, S.; Yang, H. H.; Wu, J.; and Zhang, M. 2025. MoE²: Optimizing Collaborative Inference for Edge Large Language Models. *arXiv preprint arXiv:2501.09410*.
- Kong, R.; Li, Y.; Wang, W.; Kong, L.; and Liu, Y. 2025. Serving MoE Models on Resource-Constrained Edge Devices via Dynamic Expert Swapping. *IEEE Transactions on Computers*, 74(8): 2799–2811.
- Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; and Chen, Z. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv* preprint arXiv:2006.16668.
- Li, N.; Guo, S.; Zhang, T.; Li, M.; Hong, Z.; Zhou, Q.; Yuan, X.; and Zhang, H. 2025. The MoE-Empowered Edge LLMs Deployment: Architecture, Challenges, and Opportunities. *arXiv preprint arXiv:2502.08381*.
- Liang, K.; Meng, L.; Liu, M.; Liu, Y.; Tu, W.; Wang, S.; Zhou, S.; Liu, X.; Sun, F.; and He, K. 2024. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 9456–9478.
- Lin, X.; Tian, H.; Xue, W.; Ma, L.; Cao, J.; Zhang, M.; Yu, J.; and Wang, K. 2024. Flame: Fully leveraging moe sparsity for transformer on FPGA. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 1–6.
- Liu, C.; Yu, M.; Sun, Y.; and Carlson, T. E. 2025. The Sparsity-Aware LazyGPU Architecture. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, 1020–1034.
- Liu, M.; Wang, W.; and Wu, C. 2025. Optimizing distributed deployment of mixture-of-experts model inference in serverless computing. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, 1–10. IEEE.
- Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1930–1939.
- Menghani, G. 2023. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12): 1–37.
- Pan, X.; Lin, W.; Zhang, L.; Shi, S.; Tang, Z.; Wang, R.; Li, B.; and Chu, X. 2025. Fsmoe: A flexible and scalable training system for sparse mixture-of-experts models. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 524–539.
- Petersen, F.; Borgelt, C.; Kuehne, H.; and Deussen, O. 2022. Deep differentiable logic gate networks. *Advances in Neural Information Processing Systems*, 35: 2006–2018.
- Rajbhandari, S.; Li, C.; Yao, Z.; Zhang, M.; Aminabadi, R. Y.; Awan, A. A.; Rasley, J.; and He, Y. 2022. Deepspeedmoe: Advancing mixture-of-experts inference and training

- to power next-generation ai scale. In *International conference on machine learning*, 18332–18346. PMLR.
- Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Susano Pinto, A.; Keysers, D.; and Houlsby, N. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34: 8583–8595.
- Sarkar, R.; Liang, H.; Fan, Z.; Wang, Z.; and Hao, C. 2023. Edge-moe: Memory-efficient multi-task vision transformer architecture with task-level sparsity via mixture-of-experts. In 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), 01–09. IEEE.
- Shen, L.; Sun, Y.; Yu, Z.; Ding, L.; Tian, X.; and Tao, D. 2024a. On efficient training of large-scale deep learning models. *ACM Computing Surveys*, 57(3): 1–36.
- Shen, Y.; Shao, J.; Zhang, X.; Lin, Z.; Pan, H.; Li, D.; Zhang, J.; and Letaief, K. B. 2024b. Large language models empowered autonomous edge AI for connected intelligence. *IEEE Communications Magazine*, 62(10): 140–146.
- Szatkowski, F.; Wójcik, B.; Piórczyński, M.; and Scardapane, S. 2024. Exploiting activation sparsity with dense to dynamic-k mixture-of-experts conversion. *Advances in Neural Information Processing Systems*, 37: 43245–43273.
- Tairin, S.; Mahmud, S.; Shen, H.; and Iyer, A. 2025. eMoE: Task-aware Memory Efficient Mixture-of-Experts-Based (MoE) Model Inference. *arXiv preprint arXiv:2503.06823*.
- Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In 7th International Conference on Learning Representations, ICLR.
- Wang, H.; Zhou, Q.; Hong, Z.; and Guo, S. 2025. D ² MoE: Dual Routing and Dynamic Scheduling for Efficient On-Device MoE-based LLM Serving. *arXiv* preprint *arXiv*:2504.15299.
- Yi, R.; Guo, L.; Wei, S.; Zhou, A.; Wang, S.; and Xu, M. 2025. EdgeMoE: Empowering Sparse Large Language Models on Mobile Devices. *IEEE Transactions on Mobile Computing*, 24(8): 7059–7073.
- Yin, W.; Chen, M.; Zhang, R.; Zhou, B.; Wang, F.; and Roth, D. 2024. Enhancing llm capabilities beyond scaling up. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, 1–10.
- Yu, D.; Shen, L.; Hao, H.; Gong, W.; Wu, H.; Bian, J.; Dai, L.; and Xiong, H. 2024. Moesys: A distributed and efficient mixture-of-experts training and inference system for internet services. *IEEE Transactions on Services Computing*, 17(5): 2626–2639.
- Zeng, Y.; Huang, C.; Mei, Y.; Zhang, L.; Su, T.; Ye, W.; Shi, W.; and Wang, S. 2025. EfficientMoE: Optimizing Mixture-of-Experts Model Training With Adaptive Load Balance. *IEEE Transactions on Parallel and Distributed Systems*, 36(4): 677–688.

- 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*, 1–9.
- Zhou, Y.; Lei, T.; Liu, H.; Du, N.; Huang, Y.; Zhao, V.; Dai, A. M.; Le, Q. V.; Laudon, J.; et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35: 7103–7114.