# CasMoE: A Cascaded Framework for Efficient MoE Inference on Resource-constrained Devices

**Chengcheng Wang[1,2*], Haowen He[1,2*], Liang Zhao[3], Xiaoheng Deng[4], Lixin Duan[1,2], Shaohua Wan[1,2†]**

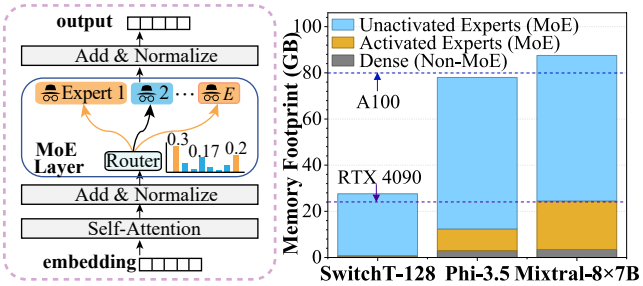
[1]University of Electronic Science and Technology of China, Chengdu, China
[2]Shenzhen Institute for Advanced Study, UESTC, Shenzhen, China
[3]School of Computer Science, Shenyang Aerospace University, Shenyang, China
[4]School of Computer Science and Engineering, Central South University, Changsha, China
wccwb1@outlook.com, {haowen_he, lxduan, shaohua.wan}@uestc.edu.cn, lzhao@sau.edu.cn, dxh@csu.edu.cn

## Abstract

The Mixture-of-Experts (MoE) architecture has emerged as a key enabler for scaling large language models (LLMs), empowering increased model capacity with minimal computational overhead through gating-based dynamic expert activation. However, due to the memory demands introduced by expert modules, MoE inference on resource-constrained devices is still challenging. Existing methods such as model compression and parameter offloading provide partial alleviation but often lead to reduced accuracy or increased latency. In this paper, we propose **CasMoE**, a general and efficient cascaded framework for accelerating MoE inference on resource-constrained devices. CasMoE employs a two-stage offline-online approach to facilitate efficient expert prefetching. In the offline stage, a parameterized Expert Activation Predictor (EAP) is introduced to accurately predict the corresponding expert activation from the incoming prompt. In the online stage, a non-parametric Expert Activation Matcher (EAM) supporting fast expert retrieval is then integrated with the EAP to form a cascade planner that operates independently of the MoE architecture, predicting activated experts for all MoE layers in a single pass prior to decoding. A gating mechanism is also incorporated to dynamically adjust the sensitivity of the EAM and EAP, enabling a flexible trade-off between inference efficiency and quality. Extensive experiments on diverse downstream tasks demonstrate CasMoE's effectiveness in accelerating inference while preserving high accuracy.

## Introduction

Large language models (LLMs) have achieved remarkable performance across a wide range of natural language processing tasks (Devlin et al. 2019; Brown et al. 2020; Achiam et al. 2023), largely driven by data and model scaling, at the cost of substantial computational overhead (Kaplan et al. 2020). The Mixture-of-Experts (MoE) architecture (Shazeer et al. 2017) has emerged as a promising solution for enhancing the scalability of LLMs by using a gating mechanism for efficient conditional computation (Fedus, Zoph, and Shazeer 2022; Jiang et al. 2024; Liu et al. 2024; Abdin et al. 2024; Bai et al. 2023), as illustrated in Figure 1a. However, the memory demands introduced by expert modules with sparse activation presents significant hurdles for inference on resource-constrained devices. For instance, as shown in Figure 1b, Mixtral-8×7B (Jiang et al. 2024) requires approximately 87GB VRAM during inference, surpassing the available memory of even high-end GPUs like the A100-80GB, which are widely used in practice. However, only about 24GB weights (i.e. dense components and activated experts) are actually used during inference.




(a) MoE Architecture (b) MoE Memory Consumption

Figure 1: MoE enables efficient conditional computation through a gating-based dynamic expert activation; however, high memory overhead introduced by expert modules with sparse activation poses significant challenges for inference on resource-constrained devices.

In recent years, model compression has become a popular research field to decrease the memory overhead by reducing the size of models (Chen et al. 2022; Kudugunta et al. 2021; Li et al. 2024). However, it often sacrifices inference accuracy, especially under the circumstance of limited resources. As an alternative, expert offloading has emerged as a promising approach, which alleviates GPU memory pressure by transferring inactive experts to the CPU. Despite its benefits, on-demand expert loading (HuggingFace 2022a) introduces significant communication overhead due to its interleaving execution between expert loading and computation. For example, when inferencing on SST-2 (Socher et al. 2013) using Mixtral-8×7B, expert loading occupies 40.8% of the to-

*These authors contributed equally.
†Corresponding author.

tal inference time, considerably degrading the inference efficiency. To mitigate the CPU→GPU expert transfer latency, SE-MoE (Shen et al. 2022) proactively pre-loads the next layer's all experts during the inference over the current one. Nevertheless, as model size increases, this method exacerbates memory usage and inference latency due to redundant expert loading, thereby limiting its applicability.

Recent work has leveraged the similarity of expert activations across adjacent MoE layers to improve inference efficiency (Eliseev and Mazur 2023; Hwang et al. 2024; Zhong et al. 2025; Tang et al. 2024; Xue et al. 2025; Yu et al. 2025). For instance, Pregated-MoE (Hwang et al. 2024) employs a pre-gating mechanism that predicts and prefetches experts of subsequent layer. HOBBIT (Tang et al. 2024) assesses expert importance and optimizes computation-communication overlap through mixed-precision expert prefetching across layers. MoE-Infinity (Xue et al. 2025) performs at the sequence level, tracking expert usage patterns based on activation similarities. Despite their benefits, fast similarity-based activation matching and static cross-layer predictor placement that requires high accuracy but lacks flexibility—often struggle to strike the balance between inference efficiency and quality, ultimately resulting in suboptimal performance on downstream tasks. Additionally, the current layer-wise prediction approach introduces the waste of pipeline parallelism capabilities, making it crucial to develop effective mitigation strategies.

To tackle the above challenges, especially the trade-off between efficient matching-based policy and effective predictor-based policy, we introduce **CasMoE**, a general and efficient cascaded framework for accelerating MoE inference on resource-constrained devices, as shown in Figure 2. CasMoE employs a two-stage offline-online approach to facilitate efficient expert prefetching. In the offline stage, given a dataset and pre-trained MoE model weights, we trace the expert activation patterns during the MoE inference over different prompts, known as data calibration. Then, we introduce a parameterized expert activation predictor (EAP) comprising a lightweight encoder and predictor heads to accurately predict the expert patterns from the prompt. Meanwhile, we construct a non-parametric expert activation matcher (EAM) which leverages similarity-based indexing over a key-value table mapping prompt embeddings to expert patterns, enabling approximate nearest expert retrieval. In the online stage, we propose a standalone Cascade Planner that integrates the EAM supporting data updates and the EAP, to predict expert patterns for all MoE layers in a single pass prior to decoding. A gating mechanism is incorporated to dynamically adjust the sensitivity of the EAM and EAP, enabling a flexible trade-off between inference efficiency and quality. Upon a new inference prompt arrives, the cascade planner operates to efficiently predict expert patterns across all MoE layers, which are then utilized to guide the top-$k$ expert selection and preloading, thereby facilitating the subsequent MoE decoding.

Our contributions are summarized as follows:

- We present CasMoE, a general and efficient cascaded framework for accelerating MoE inference on resource-constrained devices. It integrates an expert activation matcher (EAM) with an expert activation predictor (EAP), connecting with a gating mechanism that dynamically adjusts their sensitivity to enable a flexible trade-off between inference efficiency and quality.
- We introduce a non-parametric EAM that leverages similarity-based indexing over a key-value table mapping prompt embeddings to expert activation patterns, to enable approximate nearest expert retrieval while supporting data updates.
- We design a parametric EAP, comprising a lightweight encoder with contrastive learning and predictor heads architecture. The EAP is trained end-to-end to improve the accuracy of expert predictions.
- Experimental results demonstrate the effectiveness of CasMoE, achieving 65.13% throughput improvements with over 96.6% performance preservation compared to on_demand expert offloading baseline.

# Related Work

## Mixture of Experts

The MoE architecture (Shazeer et al. 2017) introduces a gating network, commonly referred to as a "router", that selectively activates only a small subset of expert modules (essentially feedforward neural networks) for each token. This mechanism enables the model to scale its capacity efficiently without a proportional increase in computational cost (Lepikhin et al. 2021; Du et al. 2022).

As illustrated in Figure 1a, a typical MoE layer consists of a router and a set of experts, denoted as $E = \{e_1, e_2, \cdots, e_K\}$. For a given token $x \in R^h$, the router first computes the activation scores over the experts:

$$G(x) := \mathrm{Softmax}(x \cdot W_g), \tag{1}$$

where $W_g \in R^{h \times K}$ is the router's weight matrix, $h$ is the dimensionality of the token representation, and $K$ is the number of experts.

In existing studies, the top-$k$ experts with the highest activation scores $G(x)$, computed based on local token-level features, are selected at each layer to participate in the forward inference. Accordingly, the output of the MoE layer can be formulated as:

$$y = \sum_{i=1}^{k} G_i(x) e_i(x), \tag{2}$$

where $G_i(x)$ is the gating score of the $i$-th selected expert, and $e_i(x)$ is the response of that expert to the input.

## Expert Offloading

Offloading inactive parameters to auxiliary storage is an effective way to alleviate GPU memory constraints and enable efficient MoE inference on resource-constrained devices. While prior work (Rajbhandari et al. 2021; Aminabadi et al. 2022; Sheng et al. 2023; Jeong, Baek, and Ahn 2023; Kwon et al. 2023) has focused on system-level inference optimization, most approaches have been tailored to dense LLMs, neglecting the sparse activation patterns intrinsic to

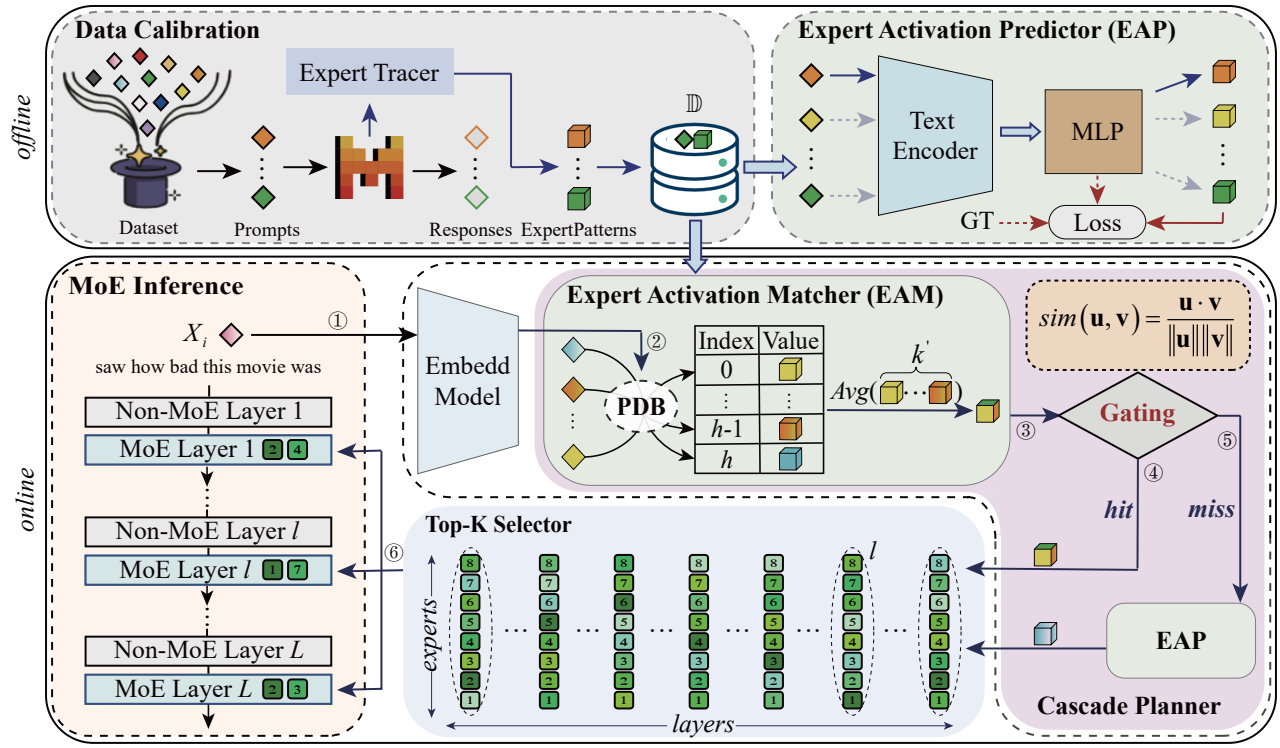



Figure 2: Overview of CasMoE. In the offline stage, we trace the expert patterns of the given dataset on the pre-trained MoE model, which are then used to construct an expert activation matcher (EAM) and train an expert activation predictor (EAP). In the online stage, we introduce a cascade planner independent of MoE that incorporates the EAM with the EAP. For each incoming prompt $X_i$, we first compute its embedding and then utilize the cascade planner to predicts the corresponding expert patterns for all MoE layers in a single pass, where a gating is used to dynamically adjust the sensitivity of the EAM and the EAP. The top-$k$ selected experts are then preloaded into the corresponding MoE layer for efficient inference.

MoE models. Another line of work (Kamahori et al. 2025; Zhang, Aggarwal, and Mitra 2025; Song et al. 2024b) explores CPU's computing capacity to assist expert execution to reduce the latency caused by expert weight transfers. However, given that the CPU's computation speed is significantly lower than that of the GPU, this approach may be less effective in latency-sensitive scenarios.

Notably, Mixtral-Offloading (Eliseev and Mazur 2023) highlights the significance of expert activation similarity across layers for expert offloading to improve the efficiency of MoE inference (Zhong et al. 2025). Building on this insight, Pregated-MoE (Hwang et al. 2024) and ProMoE (Song et al. 2024a) substitute original routers with integrated expert predictors, enabling expert prediction and prefetching at the single-layer level and across multiple strides, respectively. SiDA-MoE (Du et al. 2024) further enhances the inference efficiency by introducing a data-aware hash function that explores activation sparsity to support parallel prefetching. MoE-Infinity (Xue et al. 2025) utilizes activation similarity to track expert usage patterns at the sequence level. Meanwhile, fMoE (Yu et al. 2025) combines semantic matching with expert activation trajectories to enable cross-layer expert prefetching. However, these methods generally struggle to strike a balance between inference efficiency and quality, thereby limiting their suitability for diverse downstream tasks.

## Methodology

Figure 2 illustrates the overall framework of our CasMoE, which employs a two-stage strategy to enable a flexible trade-off between inference efficiency and quality.

### Problem Formulation

The objective of expert prefetching is two-fold: i) to obtain precise expert selections that aligns with a given prompt, and ii) to efficiently load them for MoE inference. Formally, let $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \cdots, \mathcal{E}_L\}$ denote the set of experts activated for a single token during inference, where $\mathcal{E}_l$ is the selected expert set at the $l$-th layer and $L$ is the total number of MoE layers. Consequently, $|\mathcal{E}_l|$ is typically 1 or 2. Unlike traditional token-level expert prefetching, we explore prompt-level expert prefetching, aiming to utilize effective, albeit limited, experts sized $k \times L$ for high-performance inference. $k$ is the selected number of experts in each layer. Our objective is to develop a cascaded planner $\mathcal{P}$, when given an input prompt $X_i$, anticipates the expert set $\mathcal{E}$, mathematically expressed as $\mathcal{E} = \mathcal{P}(X_i)$, to optimize inference throughput while preserving inference accuracy.

## Data Calibration

Well-designed data calibration of expert patterns is the key to successful $\mathcal{P}$. Consider a dataset comprising $M$ prompts. For each prompt $X_i = \{x_1, x_2, \cdots, x_m\}$, the pre-trained MoE model generates an output $Y_i = \{y_1, y_2, \cdots, y_n\}$ in an autoregressive manner, where $m$ and $n$ denote the lengths of the input and output, respectively. To effectively characterize our proposed prompt-level expert prefetching method, we construct a novel expert pattern label that considers both the frequency of expert selection and its significance for token decoding, which is represented by:

$$EP = \sum_{j=1}^{m} \alpha \cdot \mathbb{I}(e \in x_j) \tag{3}$$

where $\alpha$ is the output of the MoE's router and $\mathbb{I}(\cdot)$ is an indicator function, which equals 1 if $e$ is used for $x_m$ inference and 0 otherwise.

When the calibration is complete, we can obtain a expert pattern matrix $EP_i$ with the size of $L \times K$, where $L$ is the number of MoE layers and $K$ is the number of experts per layer, respectively. By pairing each prompt $X_i$ with its corresponding expert pattern $EP_i$, we obtain a collection of data pairs $\{X_i, EP_i\}$, which constitutes a dataset $\mathbb{D} = \{X_i, EP_i\}_{i=1}^{M}$.

## Cascade Planner

To balance inference efficiency and quality, we propose a cascade planner $\mathcal{P}$ that integrates a non-parametric expert activation matcher (EAM) with a parameterized expert activation predictor (EAP), enabling high-quality learning of expert activation patterns without relying on intermediate MoE results.

**Expert Activation Matcher (EAM).** To accelerate the MoE inference, it is essential to quickly identify which experts should be prefetched. To this end, we construct a non-parametric expert activation matcher (EAM) based on the dataset $\mathbb{D}$. For each prompt $X_i$, we compute its embeddings using the sentence-transformers (Reimers and Gurevych 2019), serving as the key in the Pattern Database (PDB), and store the corresponding expert activation patterns $EP_i$ as its associated value. A Hierarchical Navigable Small World (HNSW) index (Malkov and Yashunin 2018) is then built over the keys to enable similarity-based approximate nearest expert search. In particular, the distance of two prompt embeddings $\mathbf{u}$ and $\mathbf{v}$ are discriminated using cosine similarity, which is described as follows:

$$cosine\_sim\,(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|}. \tag{4}$$

In light of this, we compute the embeddings of each prompt $X_j$ in the mentioned manner to obtain the analogous expert patterns $\{EP_j\}_{j=1}^{k'}$ by retrieving the top-$k'$ most similar keys. Then, a fused output of EAM is generated by a weighted average combination of queried expert patterns, as formulated below:

$$\widehat{EP_j} = \frac{1}{k'} \sum_{j=1}^{k'} EP_j. \tag{5}$$

The $\widehat{EP_j}$ is used for final expert selection for $X_j$ inference. By leveraging the efficiency of PDB, EAM enables rapid expert retrieval.

**Expert Activation Predictor (EAP).** While improving inference speed is important, ensuring inference quality is equally vital. Comparing to EAM that offers fast patterns retrieval, the parameterized expert activation predictor (EAP) is designed to accurately predict the expert activation patterns from the prompt. As the input semantics need to be carefully comprehended to extract latent critical information from input sequences to activate the most relevant experts, EAP employs a lightweight encoder-only architecture. Specifically, EAP uses a lightweight BERT model to capture rich linguistic features within the prompt and produce a robust embedding. Following this, we attach $L$ prediction heads to model the embedding to the expert patterns. Each head is implemented as a multi-layer perceptron (MLP) with an output dimension of $K$.

**Gating.** To enable a flexible trade-off between inference efficiency and quality for different prompts, we introduce a gate to cascade planner $\mathcal{P}$ to dynamically adjust the sensitivity of the EAM and the EAP. In particular, we utilize the cosine similarity produced from EAM as a confidence score. As shown in Figure 2, if the score is higher than the threshold $\epsilon$, the result of EAM will be adopted; otherwise, EAP will be called to predict the pattern from the prompt.

## Training and Inference

**End-to-End EAP Training.** Given the dataset $\mathbb{D}$, we train the EAP in an end-to-end manner. To enhance the robustness of semantic embeddings, we adopt a unsupervised training scheme from SimCSE (Gao, Yao, and Chen 2021) to the training of our encoder. Specifically, for each prompt $X_i$, where $i \in M$, we construct its corresponding positive example $X_i^+$ by duplication, forming a positive pair $(X_i, X_i^+)$. This pair is fed into the encoder to yield two intermediate embeddings $h_i$ and $h_i^+$. The training objective for each pair $(X_i, X_i^+)$ is to minimize the InfoNCE loss:

$$\ell_i = -\log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{M} e^{sim(h_i, h_j^+)/\tau}}, \tag{6}$$

where $\tau$ is a temperature hyperparameter and $sim(h, h^+) = \frac{h^T h^+}{\|h\|\|h^+\|}$ is the cosine similarity. Accordingly, the training loss of the encoder is defined as follows:

$$\mathcal{L}_{cont} = \frac{1}{M} \sum_{i=1}^{M} \ell_i. \tag{7}$$

On the other hand, we treat this target as a regression task towards a pattern vector label. This perspective lays the groundwork for learning more flexible expert selection strategies, particularly tailored to the contiguous feature of EP. The training objective of EAP is to minimize the Huber Loss (Huber 1964) between the predicted and actual expert patterns, which is expressed as:

$$\mathcal{L}_{hb} = \begin{cases} \frac{1}{2}(p_i - \hat{p}_i)^2, & if\ |p_i - \hat{p}_i| < \delta \\ \delta(|p_i - \hat{p}_i| - \frac{1}{2}\delta), & otherwise \end{cases} \tag{8}$$

where $\hat{p}_i$ is the expert patterns predicted by the EAP, $p_i$ is the ground-truth patterns, and $\delta$ is a hyperparameter that controls the transition between mean squared error (MSE) and mean absolute error (MAE). Specifically, MSE is used when $|p_i - \hat{p}_i| < \delta$, and MAE is applied otherwise.

Therefore, the total loss function for EAP end-to-end training is described as:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{cont} + \mathcal{L}_{hb}, \tag{9}$$

where $\alpha$ is a hyperparameter.

## Inference

During the inference phase, our cascade planner $\mathcal{P}$ aims to predict the expert activation patterns for all MoE layers prior to decoding. As illustrated in Figure 2, for each incoming prompt, we first compute its embedding using embedded model and then use EAM to retrieve the top-$k'$ embeddings to derive fused patterns. EAP is then triggered for pattern prediction iff the embedding similarity is below the threshold $\epsilon$. The expert activation patterns obtained by $\mathcal{P}$ are subsequently converted to expert selection by top-$k$ and further utilized by system to prefetch corresponding experts for decoding. In addition, we introduce a buffer with size $\mathcal{C}$ to temporarily store newly arrived prompts and their corresponding expert patterns. Once the buffer reaches capacity, its contents are used to update the PDB.

# Experiments

## Experimental Setup

**Models and tasks.** We utilize Mixtral-8x7B (Jiang et al. 2024) with a mixed quantization configuration as the baseline MoE model for evaluation and conduct experiments on the general benchmark GLUE (Wang et al. 2018) and MMLU (Hendrycks et al. 2021), as well as the translation task WMT16 (Bojar et al. 2016). Specifically, we select eight datasets from the GLUE, including CoLA (Warstadt, Singh, and Bowman 2019), SST-2 (Socher et al. 2013), MRPC (Dolan and Brockett 2005), QQP (DataCanary et al. 2017), MNLI (Williams, Nangia, and Bowman 2017), QNLI (Rajpurkar et al. 2016), WNLI (Levesque, Davis, and Morgenstern 2012), and RTE (Giampiccolo et al. 2007).

**Baselines.** To evaluate the effectiveness of our proposed approach, we compare the following widely used baselines with the same model and datasets.

- **All-in-GPU**: All experts are placed on the GPU throughout inference. It is also known as "Vanilla Model".
- **MoE-OnDemand** (HuggingFace 2022a): All experts are stored on the CPU. Experts are dynamically loaded into the GPU when selected for inference.
- **SE-MoE** (Shen et al. 2022): All experts reside on the CPU. During the execution of the current MoE layer, all experts in the next-layer are pre-migrated to the GPU.
- **Mixtral-Offloading** (Eliseev and Mazur 2023): Predicts next-layer experts based on inter-layer activation similarity and employs a least recently used (LRU) strategy for expert cache replacement.
- **AdapMoE** (Zhong et al. 2025): A sensitivity-based adaptive gating mechanism is utilized to enable the prefetching of multiple next's layer experts.

**Implementation.** We implement CasMoE based on PyTorch. Given the nascent nature of our research focus, we design a representation of prompt-level expert activation patterns. EAM uses a buffer size of $\mathcal{C} = 30$ to refresh the PDB with new datas. EAP uses six-layer TinyBERT and a two-layer MLP network, with a hidden size of 64 and an intermediate size of 128 in most scenarios. Adam (Kingma and Ba 2015) optimizer is used for training EAP with a learning rate of $6 \times 10^{-7}$ and a loss weight of 0.05. All experiments are conducted on a server equipped with four NVIDIA RTX 4090 GPUs and dual Intel Xeon Gold 6330 CPUs.

**Evaluation Metrics.** Similar to existing research on expert offloading, we quantitatively compare the performance of MoE inference in terms of quality and efficiency, using the HuggingFace's Evaluate library (HuggingFace 2022b). Specifically, we adopt standard evaluation metrics: SacreBLEU for WMT16 and accuracy for GLUE and MMLU benchmarks. Note that higher accuracy, SacreBLEU and throughput, as well as lower inference latency, reflect better performance.

## Experiment Results

**Inference Quality.** Table 1 provides the inference quality comparisons between CasMoE and baselines on GLUE. We use All-in-GPU as the baseline to indicate the upper bound of performance among all methods. As shown in the table 1, All-in-GPU consistently achieves top results across most metrics, supporting its role as the performance upper bound. Since all needed experts are scheduled precisely, both MoE-OnDemand and SE-MoE achieves the same inference accuracy as the All-in-GPU on all datasets. Among all predictive work, our method is commonly competitive over most datasets, performing as the second-best result in all baselines. We observe that our method yields suboptimal performance on certain datasets like CoLA and QQP, which are characterized by high subjectivity. After quantization and offloading, CasMoE generative capability is somewhat weakened, leading to reduced comprehension of such complex and nuanced inputs.

Table 2 illustrates the accuracy comparison on MMLU and WMT16. Results demonstrates that Mixtral-Offloading and AdapMoE are able to maintain relatively good performance on MMLU, but perform poorly on long-sequence tasks. The former and the latter can only reach 63% and 57% of vanilla model's performance. In contrast, our CasMoE achieves inference performance of 0.572 and 34.153 on MMLU and WMT16, preserving approximately 57.2% and 80% of the inference accuracy, respectively. This benefits from our high-quality expert prediction and prefetching at prompt level, which jointly take into account both the frequency of expert selection and its significance to token decoding.

**Inference Efficiency.** Figure 5 presents the throughput and latency comparisons on GLUE. In contrast to previ-

| Method | GLUE | | | | | | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| | CoLA | SST-2 | MRPC | QQP | MNLI | QNLI | WNLI | RTE | |
| All-in-GPU | **0.72** | **0.89** | **0.74** | 0.72 | **0.54** | **0.74** | **0.69** | **0.77** | **0.73** |
| MoE-OnDemand | 0.72 | 0.89 | 0.74 | 0.72 | 0.54 | 0.74 | 0.69 | 0.77 | 0.73 |
| SE-MoE | 0.72 | 0.89 | 0.74 | 0.72 | 0.54 | 0.74 | 0.69 | 0.77 | 0.73 |
| Mixtral-Offloading | 0.64 | 0.69 | 0.51 | 0.59 | 0.48 | 0.54 | 0.62 | 0.62 | 0.59 |
| AdapMoE | 0.72 | 0.88 | 0.67 | **0.78** | 0.39 | 0.48 | 0.56 | 0.68 | 0.64 |
| CasMoE (ours) | 0.72 | 0.88 | 0.72 | 0.70 | 0.50 | 0.73 | 0.69 | 0.74 | 0.71 |

Table 1: We evaluated the inference accuracy of our method against the most representative approaches on GLUE, where our method consistently outperformed other expert prefetching works on most of datasets. **Bold** values indicate the best results. Underline values indicate the second-best results.

| Method | QA.Task | Trans.Task |
|---|---|---|
| | MMLU | WMT16 |
| All-in-GPU | **0.792** | **43.366** |
| MoE-OnDemand | 0.792 | 43.366 |
| SE-MoE | 0.792 | 43.366 |
| Mixtral-Offloading | 0.626 | 27.720 |
| AdapMoE | 0.507* | 25.024 |
| CasMoE (ours) | 0.572 | 34.153 |

Table 2: Accuracy comprisons on MMLU and WMT16. * indicates OOM occurred with all prompts, so the evaluation entries are slightly reduced. Underline values indicate the second-best results.

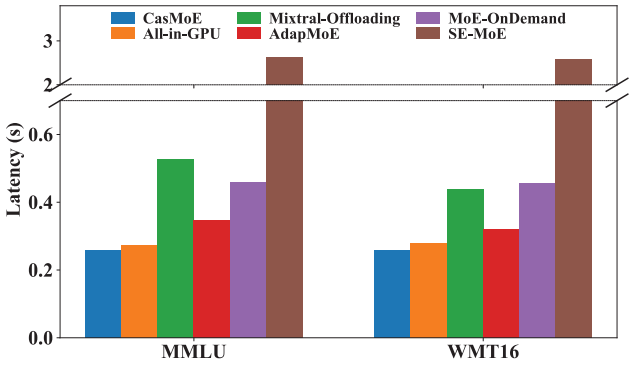



Figure 3: Latency evaluation on MMLU and WMT16.

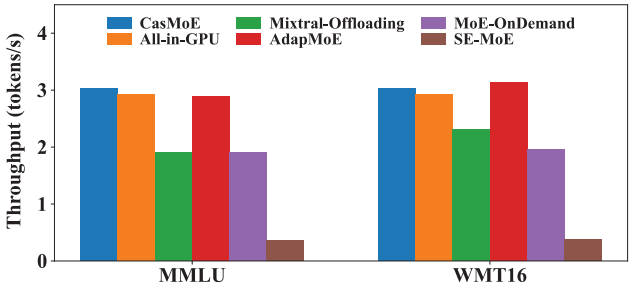



Figure 4: Throughput evaluation on MMLU and WMT16.

ous baselines, CasMoE delivers a 10× performance improvement over SE-MoE and a 2× improvement over ME-OnDemand. Furthermore, our prompt-level design enables CasMoE to achieve notable increasement in inference efficiency over original model, even in comparison with other baselines with predictor designs. As evidenced by results, a throughput exceeding 3.0 is achieved for nearly all GLUE subsets.

We also illustrate the inference efficiency of CasMoE on MMLU and WMT16, as shown in Figure 3 and Figure 4. The results highlight that CasMoE achieves higher-quality output by activating only a small set of frequently effective experts, outperforming token-based selection in extended sequence inference while maintaining high efficiency. This demonstrates CasMoE's capability to effectively capture expert activation patterns.

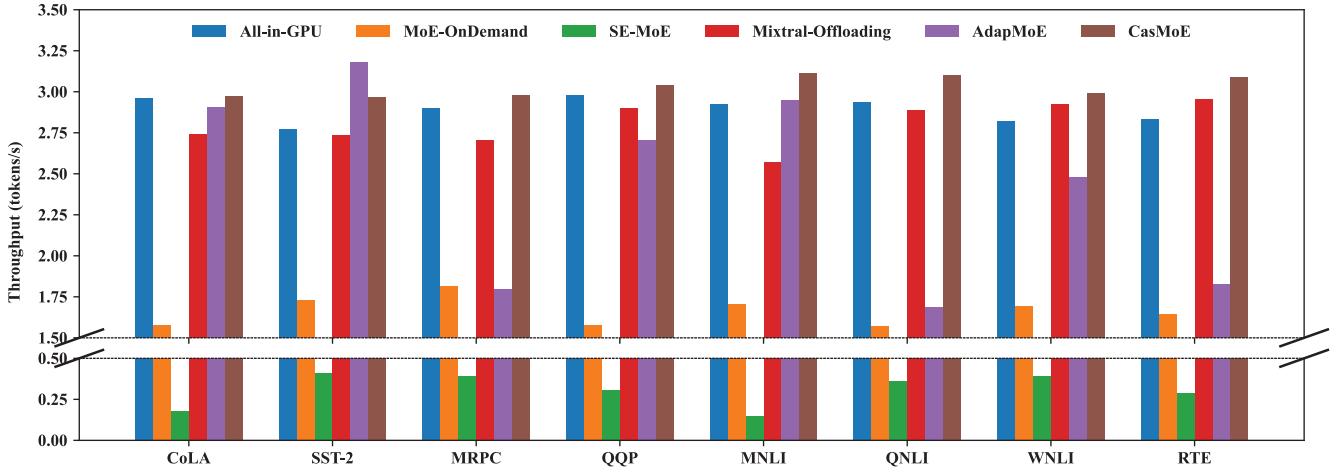
| Method | Accuracy$^{\uparrow}$ | Throughput$^{\uparrow}$ |
|---|---|---|
| All-in-GPU | 0.72 | 2.963 |
| ours (EAP) | 0.71 | 2.971 |
| ours (EAP + EAM) | **0.72** | **3.074** |

Table 3: Inference quality and efficiency results of EAP, EAP with EAM ($\epsilon = 0.3$) and All-in-GPU on CoLA.

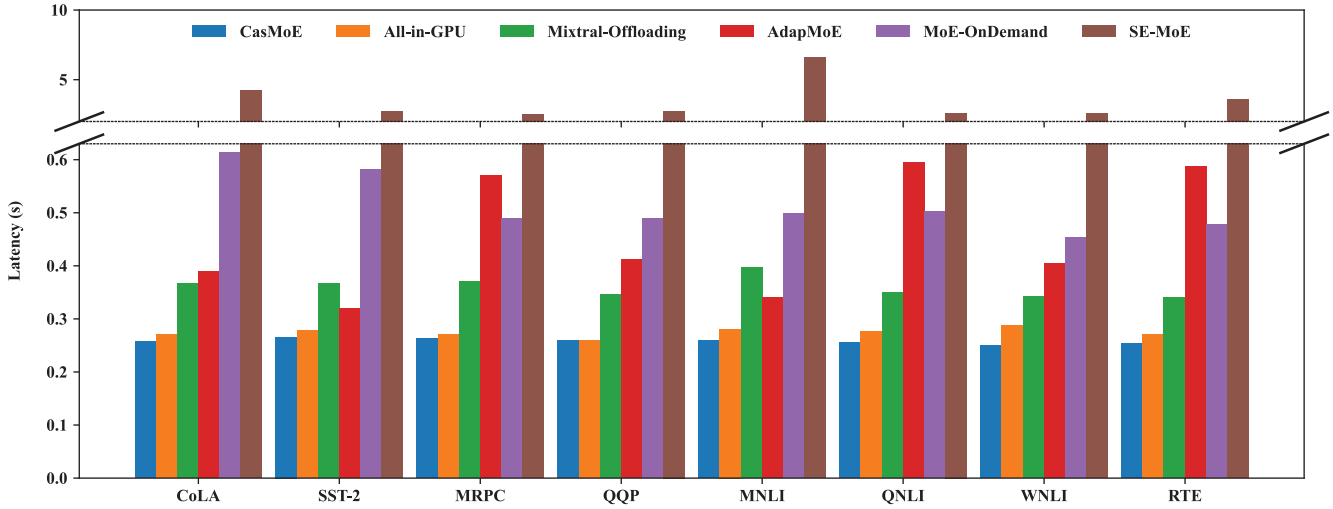
| Label | Accuracy$^{\uparrow}$ |
|---|---|
| All-in-GPU | 0.72 |
| count | 0.62 |
| ours (EP) | **0.72** |

Table 4: Evaluation of label design on CoLA. Note that "count" and "All-in-GPU" denotes prompt-level expert frequency softmax and token-level expert selection, respectively.

**Ablation Study.** As illustrated in Table 3, EAP shows strong performance, achieving 98% prediction accuracy




(a) Throughput




(b) Latency

Figure 5: Throughput and latency comparisons of CasMoE with different baselines on GLUE.

with 10.73 ms computation and 300 MB memory. Combining EAP with EAM further leads to even better performance, as EAM alone attains comparable prediction accuracy with only 1.22 ms computation. This improvement stems from EAP's data collection, we directly gathered EAP's outputs, introducing additional information. By aggregating multiple outputs, we adopted a strategy similar to few-shot learning. Moreover, EAM's fast retrieval allow us to bypass BERT computation, reducing planner overhead and improving throughput.

Table 4 indicates the effectiveness of our prompt-level pattern scheme. Due to unstable distribution of collected label using count and difficulty for EAP to predict, its inference accuracy is relatively low. In contrast, our EP's design fully considers experts' influence during inference. Moreover, the EP matrix is smoother, making it more amenable to gradient-based prediction. As a result, EP outperforms the count-based softmax approach.

## Conclusion

In this paper, we present CasMoE, a novel dynamic inference acceleration framework designed to enhance MoE inference performance on resource-limited devices. CasMoE introduces a cascade planner, which performs to efficiently predict expert activations of all MoE layers. This enables expert selection and preloading to facilitate decoding. Considering both inference efficiency and quality are vital for performance, the planner integrates two components: EAP for precise expert pattern prediction and EAM for rapid similarity-based retrieval. A gating mechanism dynamically adjusts the influence of EAM and EAP to balance inference efficiency and quality flexibly. Extensive experiments on widely used datasets demonstrate that CasMoE achieves state-of-the-art performance across a range of tasks.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62172438, Shenzhen Science and Technology Program JCYJ20220818103200002, JCYJ20240813114223031 and the Natural Science Foundation of Guangdong Province under Grant. No.2024A1515011155.

## References

Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A. A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aminabadi, R. Y.; Rajbhandari, S.; Awan, A. A.; Li, C.; Li, D.; Zheng, E.; Ruwase, O.; Smith, S.; Zhang, M.; Rasley, J.; et al. 2022. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15. IEEE.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Bojar, O.; Buck, C.; Chatterjee, R.; Federmann, C.; Guillou, L.; Haddow, B.; Huck, M.; Yepes, A. J.; Névéol, A.; Neves, M.; Pecina, P.; Popel, M.; Koehn, P.; Monz, C.; Negri, M.; Post, M.; Specia, L.; Verspoor, K.; Tiedemann, J.; and Turchi, M., eds. 2016. *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Berlin, Germany: Association for Computational Linguistics.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, T.; Huang, S.; Xie, Y.; Jiao, B.; Jiang, D.; Zhou, H.; Li, J.; and Wei, F. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

DataCanary; hilfialkaff; Jiang, L.; Risdal, M.; Dandekar, N.; and tomtung. 2017. Quora Question Pairs. https://kaggle.com/competitions/quora-question-pairs. Kaggle.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

Dolan, B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Du, N.; Huang, Y.; Dai, A. M.; Tong, S.; Lepikhin, D.; Xu, Y.; Krikun, M.; Zhou, Y.; Yu, A. W.; Firat, O.; et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, 5547–5569. PMLR.

Du, Z.; Li, S.; Wu, Y.; Jiang, X.; Sun, J.; Zheng, Q.; Wu, Y.; Li, A.; Li, H.; and Chen, Y. 2024. Sida: Sparsity-inspired data-aware serving for efficient and scalable large mixture-of-experts models. *Proceedings of Machine Learning and Systems*, 6: 224–238.

Eliseev, A.; and Mazur, D. 2023. Fast inference of mixture-of-experts language models with offloading. *arXiv preprint arXiv:2312.17238*.

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.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Giampiccolo, D.; Magnini, B.; Dagan, I.; and Dolan, B. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In Sekine, S.; Inui, K.; Dagan, I.; Dolan, B.; Giampiccolo, D.; and Magnini, B., eds., *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 1–9. Prague: Association for Computational Linguistics.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Huber, P. J. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101.

HuggingFace. 2022a. HuggingFace Accelerate. https://huggingface.co/docs/accelerate/index.

HuggingFace. 2022b. HuggingFace Evaluate. https://huggingface.co/docs/evaluate/index.

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.

Jeong, J.; Baek, S.; and Ahn, J. 2023. Fast and efficient model serving using multi-gpus with direct-host-access. In *Proceedings of the Eighteenth European Conference on Computer Systems*, 249–265.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kamahori, K.; Tang, T.; Gu, Y.; Zhu, K.; and Kasikci, B. 2025. Fiddler: CPU-GPU Orchestration for Fast Inference of Mixture-of-Experts Models. In *The Thirteenth International Conference on Learning Representations*.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and

Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kudugunta, S.; Huang, Y.; Bapna, A.; Krikun, M.; Lepikhin, D.; Luong, M.-T.; and Firat, O. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J.; Zhang, H.; and Stoica, I. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.

Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; and Chen, Z. 2021. {GS}hard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *International Conference on Learning Representations*.

Levesque, H. J.; Davis, E.; and Morgenstern, L. 2012. The Winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, 552–561. AAAI Press. ISBN 9781577355601.

Li, P.; Zhang, Z.; Yadav, P.; Sung, Y.-L.; Cheng, Y.; Bansal, M.; and Chen, T. 2024. Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy. In *The Twelfth International Conference on Learning Representations*.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Malkov, Y. A.; and Yashunin, D. A. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4): 824–836.

Rajbhandari, S.; Ruwase, O.; Rasley, J.; Smith, S.; and He, Y. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 1–14.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Su, J.; Duh, K.; and Carreras, X., eds., *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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. In *International Conference on Learning Representations*.

Shen, L.; Wu, Z.; Gong, W.; Hao, H.; Bai, Y.; Wu, H.; Wu, X.; Bian, J.; Xiong, H.; Yu, D.; et al. 2022. Se-moe: A scalable and efficient mixture-of-experts distributed training and inference system. *arXiv e-prints*, arXiv–2205.

Sheng, Y.; Zheng, L.; Yuan, B.; Li, Z.; Ryabinin, M.; Chen, B.; Liang, P.; Ré, C.; Stoica, I.; and Zhang, C. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, 31094–31116. PMLR.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Song, X.; Zhong, Z.; Chen, R.; and Chen, H. 2024a. Promoe: Fast moe-based llm serving using proactive caching. *arXiv preprint arXiv:2410.22134*.

Song, Y.; Mi, Z.; Xie, H.; and Chen, H. 2024b. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, 590–606.

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 preprint arXiv:2411.01433*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In Linzen, T.; Chrupała, G.; and Alishahi, A., eds., *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. Brussels, Belgium: Association for Computational Linguistics.

Warstadt, A.; Singh, A.; and Bowman, S. R. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7: 625–641.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

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.

Yu, H.; Cui, X.; Zhang, H.; and Wang, H. 2025. fMoE: Fine-Grained Expert Offloading for Large Mixture-of-Experts Serving. *arXiv preprint arXiv:2502.05370*.

Zhang, Y.; Aggarwal, S.; and Mitra, T. 2025. DAOP: Data-Aware Offloading and Predictive Pre-Calculation for Efficient MoE Inference. In *2025 Design, Automation & Test in Europe Conference (DATE)*, 1–7. IEEE.

Zhong, S.; Liang, L.; Wang, Y.; Wang, R.; Huang, R.; and Li, M. 2025. 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. Association for Computing Machinery. ISBN 9798400710773.