MMEC: Using MoE Efficiently for Mobile Edge Computing

Sakhawat Hossan Jing Deng
Dept. of Computer Science, UNC Greensboro, Greensboro, NC, U. S. A.
s_hossan@uncg.edu, jing.deng@uncg.edu

***Abstract*—Mixture of Experts (MoE) is gaining increased attention in the field of Large Language Model (LLM) research due to its ability to enhance model scalability. This technique is particularly advantageous for devices with limited resources, such as mobile edge devices. However, identifying suitable pre-trained off-the-shelf MoE models can be challenging, requiring fine-tuning for specific tasks at hand. However, fine-tuning MoE is a complex and time-consuming process that demands significant resources, making it impractical for mobile edge devices. In this work, we propose a novel framework to assist mobile edge devices in identifying the optimal fine-tuned or off-the-shelf MoE models required for specific tasks. Our framework, termed MoE for Mobile Edge Computing (MMEC), maintains a repository of MoE models fine-tuned with varying numbers of experts on popular datasets and some general-purpose pre-trained LLMs. It analyzes a small number of randomly selected sampled queries from mobile edge devices before recommending suitable MoE models. Through extensive simulations, we demonstrate that MMEC achieves strong F1 scores with lower computation times and maintains a healthy overall gain, defined as a combination of different performance metrics, over other techniques.**

***Index Terms*—Mixture of Experts, Large Language Models, LLM, MoE, Mobile Edge Devices, Resource Optimization**

I. Introduction

Generative Artificial Intelligence (GAI) is revolutionizing applications such as intelligent chatbots, translation services, and customer support. Large Language Models (LLMs) are pivotal to this transformation, enabling sophisticated natural language processing tasks. The Mixture of Experts (MoE) model is a critical innovation that enhances scalability by using multiple expert networks, each specializing in different data subsets, to improve inferencing performance in sometimes heterogeneous tasks [1].

Deploying LLMs on edge devices, which are limited in computational and memory resources, presents significant challenges. Traditional models are often too resource-intensive for edge deployment. MoE models, which activate a subset of experts during inference, offer a promising solution. However, fine-tuning and identifying pre-trained MoE models for specific tasks is complex and demands resources beyond what edge devices can typically provide [2].

The computational demands of MoE models require efficient resource management strategies. Dynamic expert selection and load balancing can help optimize the utilization of edge devices' limited resources [3]. Furthermore, hardware accelerators like GPUs and TPUs enhance the viability of MoE models for delay-critical applications [4]. The integration of MoE models with edge computing environment opens new possibilities for applications in IoT, autonomous vehicles, and smart healthcare systems, where low latency and high reliability are essential [5]. Current methods for optimizing LLM deployment on edge devices, such as memory swapping or expert pruning, often result in trade-offs between memory usage and inference performance [6], [7].

In order to address these challenges, we propose an MMEC, MoE for Mobile Edge Computing, framework. MMEC maintains a repository of MoE models fine-tuned with varying numbers of experts and recommends suitable models based on the specific task and resource constraints. Extensive simulations show that MMEC achieves higher F1 scores with lower execution time and CPU usage, with only a slight increase in memory consumption. This approach significantly improves the feasibility of deploying LLMs on edge devices without compromising performance. In our design, two key elements are used: Dynamic Expert Activation and Softmax-based Gating Mechanism.

The rest of the paper is structured as follows: Section II provides a review of related work. In Section III, we introduce the details of our proposed framework. Section IV evaluates the proposed framework under various settings. Finally, Section V concludes the paper and discusses potential directions for future work.

II. Background and Related Work

The concept of LLMs originated from early natural language processing (NLP) methods such as n-gram and hidden Markov models, which were limited by predefined vocabularies and simplistic probabilistic frameworks [8]. Neural network-based models, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, marked a significant leap forward by enabling the modeling of sequences and capturing longer dependencies in text [9]. These models allowed for more accurate predictions and laid the groundwork for modern LLMs.

The development of LLMs accelerated with the introduction of the Transformer architecture by Vaswani et al. in 2017, which revolutionized NLP by addressing the limitations of RNNs and LSTMs through self-attention mechanisms [10]. This breakthrough enabled parallel computation and scalable

models, leading to the development of influential models like BERT [11], DeBERTa, GPT [12], and T5 [13]. BERT's bidirectional context improved NLP tasks, while GPT demonstrated powerful results in text generation through pre-training and fine-tuning.

MoE was initially proposed by Jacobs et al. in the early 1990s, featuring multiple expert models with a gating network to select the appropriate expert(s) [14]. Despite its early promise, MoE models were hampered by scalability challenges due to high computational requirements. Recent advances, particularly in the Transformer architecture and hardware, have revitalized interest in MoE models, integrating them into the Sparsely Gated Mixture-of-Experts layer to train large networks efficiently by activating only a subset of experts [1].

Recent innovations in MoE models have focused on efficiency and scalability. Switch Transformers, introduced by Wang et al. [2], scaled to trillion-parameter models using sparsity techniques, achieving significant performance improvements with reduced computational costs. Dynamic expert selection and load-balancing techniques have further optimized resource utilization, making MoE models more practical for real-time applications on resource-constrained devices.

The deployment of LLMs on edge devices poses significant challenges due to their demands for memory and computational power. Zhang et al. [15] proposed a framework called EdgeShard, which leverages collaborative edge computing to address these challenges. By distributing the computation across geo-distributed edge devices and cloud servers, EdgeShard optimizes memory usage and computational load. The framework includes three key stages: profiling, scheduling optimization, and collaborative inference. It minimizes inference latency and maximizes throughput, making it feasible to run LLMs on resource-constrained edge devices. However, the reliance on collaborative edge computing introduces extra complexities in managing distributed resources and ensuring data privacy.

Our proposed framework offers several distinct advantages over existing solutions:

- Edge Device Task Analyzer: This component analyzes a small sample of the potential list of tasks at hand close to the edge device, tailoring model deployment for better resource usage.
- Library of Pretrained MoE Models and Off-the-Shelf LLM Models: By maintaining a comprehensive repository of fine-tuned MoE models and general-purpose LLMs, the framework can quickly identify the most suitable model for a given task without the need for complete re-trainings.
- Optimal Model Suggestion: The framework intelligently suggests the best MoE or general LLM models for specific tasks, balancing inference accuracy, response time, memory usage, and computational cost.

III. The MMEC Framework

The proposed architecture, depicted in Figure 1, is designed to leverage the scalability of MoE models in edge devices, optimizing both response accuracy and efficiency. The architecture consists of three primary components: the pool of edge devices, the MMEC edge client, and the MMEC cloud infrastructure. This distributed framework allows the system to dynamically select and fine-tune models based on the specific capabilities of the edge device and the task at hand, ultimately enhancing model performance.

A. The pool of edge devices

The pool of edge devices represents a diverse set of connected devices, e.g., a WiFi router serving as Edge Device 1 and a UAV serving as Edge Device 2, that interact with the edge client. The onboard edge client samples the computational tasks, along with the device's hardware specifications and profile information. Before querying the model library, the client assembles a compact request that includes the task signature (domain, estimated input size/tokens, latency–quality targets), device profile (CPU/GPU/NPU type, RAM/VRAM, battery, current load), network status, a digest of locally cached experts, policy constraints (e.g., privacy/offline), and a tentative expert-budget range. Through the Task Analyzer, the client selects the most suitable MoE configuration tailored to the device and task requirements. This process ensures the model activates only the necessary experts without overburdening limited edge resources, balancing response quality and efficiency.

B. Pre-trained model library

The pre-trained MoE Library supplies domain-tuned

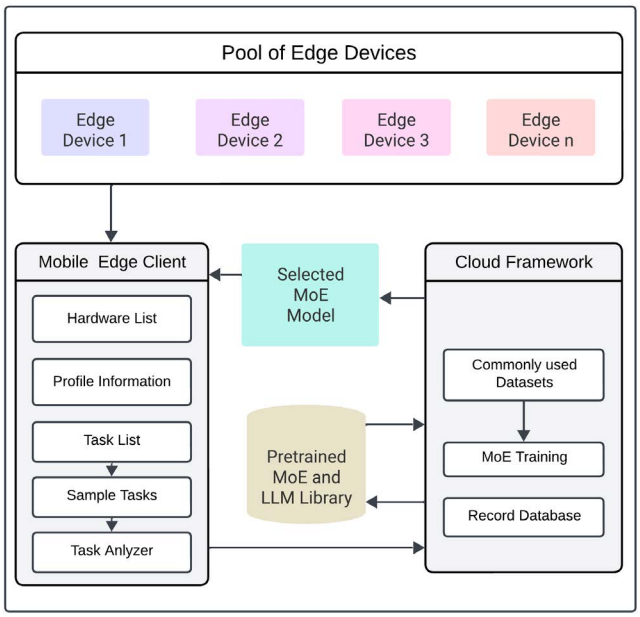

Fig. 1: MMEC framework architecture

models and activates only a minimal subset of experts per task to reduce memory and compute footprint. In response to a client request, the MMEC Cloud consults performance logs to recommend the expert count k and routing (i.e., the component that directs tokens to the most suitable experts), and returns a compact package containing model_id, quantization level, expert IDs, gating thresholds, token/latency budget, and cache policy. When bandwidth is limited, only the required experts are transmitted; if offline, the client falls back to a local heuristic. The cloud also hosts training pipelines, datasets, and results storage, enabling continuous improvements that are pushed back to edge devices.

C. Task Analyzer and Fine-Tuning Process

Figure 2 demonstrates the fine-tuning workflow managed by the Task Analyzer. This component is central for optimizing MoE model performance by analyzing various datasets, such as SQuAD V2.0 and MedMCQA. The process begins by splitting each dataset into training, validation, and testing sets to ensure robust model evaluation. After splitting the data, the trainer (for example, a BERT model) is initialized and subsequently fine-tuned on the specific dataset. This fine-tuning process allows the MoE model to adapt to the nuances of each dataset, ensuring that the model performs optimally across different tasks.

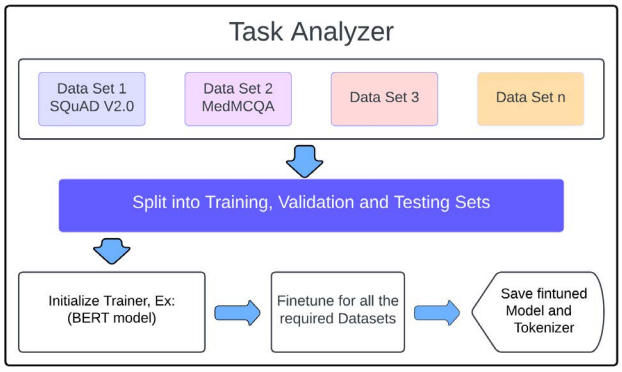

Fig. 2: MMEC Task analyzer fine tuning

The final step involves saving the fine-tuned model and tokenizer for future deployments on edge devices. By continuously fine-tuning models for different datasets, the Task Analyzer ensures that edge devices receive highly specialized models tailored to the types of questions or tasks they are likely to encounter. This modular approach to model fine-tuning enhances both the accuracy and adaptability of the system.

D. MoE model architecture

The architecture presented in this article integrates an MoE framework into a BERT model for a question-answering task. The base architecture is enhanced with a custom MoE design that leverages multiple expert modules. Specifically, the architecture introduces a set of linear layers (experts) that are selectively activated during inference based on a gating mechanism. This process, known as *expert routing*, enables the model to dynamically determine which subset of experts should be used for each input. The gating mechanism, a linear layer followed by softmax, produces scores that dictate how much each expert contributes to the output. By utilizing a dropout layer for regularization and selectively applying different experts, the model increases its capacity to handle diverse types of inputs effectively.

A unique design aspect of this architecture is its ability to scale dynamically with the number of experts without uniformly increasing computational cost, as only a subset of experts is activated for each input. This allows the model to achieve a higher level of specialization and capacity while maintaining efficient resource usage. The expert routing mechanism helps improve performance and reduce overfitting by guiding inputs to the most relevant experts. This dynamic and selective activation contributes to both improved F1 scores and reduced response generation time. I

IV. Performance Evaluation

A. Experiment Setup

Platforms: To emulate a heterogeneous edge computing environment, we utilized a Python virtual environment to simulate multiple edge devices with varying configurations. Each virtual edge node operated in isolation, allowing independent execution and result collection. The entire simulation, including both the central server and all emulated edge devices, was conducted virtually on a high-performance Linux server. This approach enabled controlled experimentation of distributed inference scenarios in a scalable and reproducible manner.

Tasks, Datasets, and Models: We employed two distinct question-answering datasets, SQuAD V2.0 [16] and MedMCQA [17], for our experiments. SQuAD V2.0 is a widely used dataset for general-purpose question answering, consisting of over 100,000 answerable questions and 50,000 unanswerable ones. In contrast, MedMCQA is a large-scale, multiple-choice dataset specifically curated for medical domain question-answering tasks.

For modeling, we developed MoE architectures based on both the Base BERT and Base DeBERTa models. We experimented with varying the number of experts from 1 to 32, with the goal of generating contextually aligned answers to the reference responses in the datasets. All MoE models were fine-tuned for 2 and 5 epochs to assess generalization and convergence. By employing two diverse datasets and two distinct language models, we aim to demonstrate the robustness and generalizability of our MoE-based framework across domains and model architectures.

Baselines: As baselines, we utilized the original Base BERT and Base DeBERTa models without expert routing.

The performance of these baselines was compared against their corresponding MoE variants, evaluating improvements in answer accuracy and generation efficiency.

Evaluation Metrics: We evaluate the overall performance of our proposed framework using multiple quantitative metrics:

1) **Download size:** the total size of the models that need to be downloaded onto mobile edge devices.
2) **Training time:** the amount of time required to train each model.
3) **Response Quality Evaluation:** To assess the quality of the generated answers, we employ two widely adopted evaluation methods: **BERTScore** [18], which calculates the semantic similarity between candidate and reference answers using contextual embeddings, and the **SQuAD Evaluation** [16], which reports exact match and F1 scores based on ground-truth answers.
4) **Gain of MoE Selection:** computed using analyzer confidence of expert selection δ, response F1 score, analyzing time efficiency η_a (defined as the analyzing time divided by the maximum analyzing time), and response time efficiency η_r (defined as the response generation time divided by the maximum response generation time)[1]

$$G = \delta + F - \eta_a - \eta_r \tag{1}$$

These combined metrics provide a comprehensive understanding of both system-level performance and the semantic fidelity of generated responses.

TABLE I: Download size for different MoE models (0 means base and size increases shown as MB for each additional expert)

# of Experts	0	1	2	4	8	16	32
BERT dl. size	417	419	422	426	435	453	489
Size Increase	N/A	2	3	2	2.25	2.25	2.25
DeBERTa dl. size	371	388	390	394	404	423	458
Size Increase	N/A	17	19	23	33	52	87

B. Download size and training time

Table I presents the download sizes for MoE configurations built on both BERT and DeBERTa models. The baseline BERT model is 417 MB, with download size gradually increasing to 489 MB as the number of experts reaches 32—an overall increase of 72 MB (about 17%). Each additional expert contributes roughly 2–3 MB. In contrast, the DeBERTa baseline starts smaller at 371 MB, but its MoE variants scale less efficiently, reaching 458 MB with 32 experts. The per-expert size increase ranges from 17 MB to 87 MB, resulting in a 23% growth. These results highlight that while DeBERTa offers a smaller starting footprint, BERT-based MoE models scale more efficiently in terms of download size, an important consideration for edge deployment scenarios.

[1]Other more sophisticated measurements of combining this group of metrics are obviously possible, as heavier focus can be placed on response quality (F1 score), response latency η_r, or analyzing time efficiency η_a, and will be investigated in our future work.

TABLE II: Training time (in hours) for different MoE models (BERT and DeBERTa), datasets (SQuAD and MedMCQA), and epoch numbers (2 and 5) under different numbers of experts (n)

Ep.	Model–Dataset	n=1	n=4	n=8	n=16	n=32
2	BERT–SQuAD	0.57	0.58	0.61	0.68	0.74
2	BERT–MedMCQA	0.58	0.60	0.66	0.71	0.77
2	DeBERTa–SQuAD	0.47	0.53	0.60	0.64	0.71
2	DeBERTa–MedMCQA	0.52	0.61	0.69	0.76	0.88
5	BERT–SQuAD	1.45	1.59	1.62	1.73	1.92
5	BERT–MedMCQA	1.45	1.48	1.53	1.67	1.87
5	DeBERTa–SQuAD	1.05	1.19	1.26	1.32	1.43
5	DeBERTa–MedMCQA	1.21	1.35	1.41	1.54	1.63

In Table II, we report the training times (in hours) for both BERT and DeBERTa-based MoE models as the number of experts increases. Results are shown for two datasets (SQuAD and MedMCQA) and two training schedules (2 and 5 epochs). As expected, training time increases with the number of experts due to the added computational load. Increasing the number of epochs from 2 to 5 results in approximately 2.5× longer training durations across all configurations. Notably, DeBERTa-based models generally require less training time than their BERT counterparts under similar conditions.

C. Response Quality Evaluation

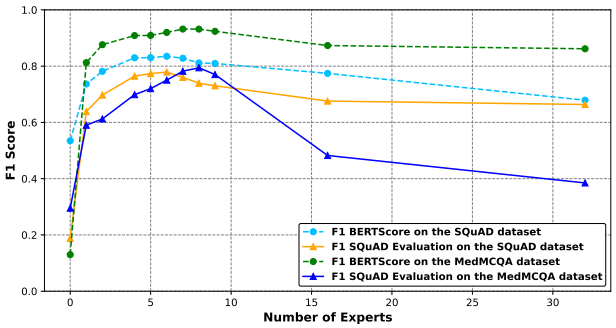

Fig. 3: BERT: F1 score for BERT Scorer and SQuAD Evaluation

Figures 3 and 4 present the F1 score trends for BERT-based and DeBERTa-based MoE models, respectively, across the SQuAD and MedMCQA datasets. In both cases, the use of multiple experts consistently improves model performance over the base models. Notably, DeBERTa-based models demonstrate a more pronounced improvement in the medical domain (MedMCQA), particularly at higher expert counts. These results reinforce the effectiveness of the MoE approach and suggest that the benefits of expert-based architectures generalize across model types and application domains.

TABLE III: SQuAD V2.0 evaluation score across datasets, question types, and number of experts (n).

Dataset	Model	Q. Type	n=0	n=1	n=2	n=4	n=5	n=6	n=7	n=8	n=9	n=16	n=32
SQuAD	BERT	Ans.	0.468	0.5577	0.5752	0.7291	0.7479	**0.7582**	0.7267	0.6951	0.6732	0.5298	0.5287
SQuAD	BERT	Unans.	0.3295	0.7106	0.8058	0.8105	0.8284	**0.8309**	0.8023	0.7967	0.7832	0.8059	0.7839
SQuAD	DeBERTa	Ans.	0.4189	0.488	0.5439	0.6537	0.683	0.6992	0.7202	**0.7603**	0.7593	0.7029	0.6732
SQuAD	DeBERTa	Unans.	0.1939	0.2393	0.2939	0.3809	0.4219	0.4559	0.4294	0.4602	**0.4782**	0.4292	0.4203
MedMCQA	BERT	Ans.	0.3448	0.644	0.7424	0.7834	0.8036	0.8275	0.8379	**0.8553**	0.8289	0.5491	0.4184
MedMCQA	BERT	Unans.	0.2462	0.5529	0.4871	0.5494	0.6298	0.682	0.7182	**0.7307**	0.7198	0.4241	0.3532
MedMCQA	DeBERTa	Ans.	0.7167	0.8804	0.894	0.9208	0.9383	0.9272	0.9488	**0.9551**	0.9477	0.9234	0.8298
MedMCQA	DeBERTa	Unans.	0.4802	0.7833	0.8183	0.8492	0.8955	0.9202	0.9272	**0.9392**	0.9206	0.8723	0.8203

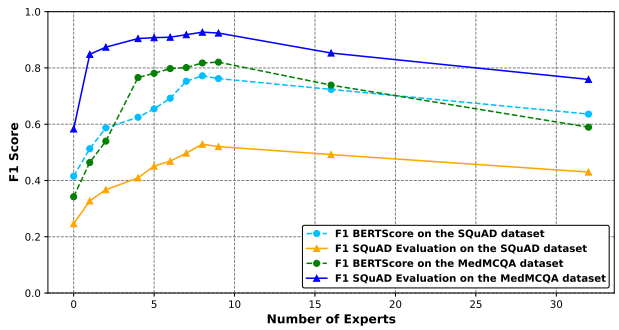

Fig. 4: DeBERTa: F1 score for BERT Scorer and SQuAD Evaluation

D. SQuAD V2 Evaluation

Table III presents the SQuAD evaluation results, reporting F1 scores separately for answerable and unanswerable questions across different expert counts. The results cover both BERT and DeBERTa models, evaluated on the SQuAD and MedMCQA datasets. Across all settings, the MoE architecture consistently outperforms the corresponding base models. Peak performance is generally achieved at intermediate expert configurations (typically between 6 and 8 experts), beyond which diminishing returns or slight declines are observed.

These findings reinforce the benefits of expert-based specialization and support the robustness of the MoE framework across different model types, question types, and domains. The results also suggest that moderate expert sizes strike a practical balance between performance and computational efficiency, with the optimum $n = 8$ for many cases, except a few at 6 or 9.

E. Response Generation Time

Table IV reports the response generation times for BERT and DeBERTa models with varying numbers of experts, evaluated on both the SQuAD and MedMCQA datasets. Across all settings, the base models (0 experts) exhibit significantly longer response times compared to their MoE counterparts. Introducing expert routing leads to substantial reductions in latency, especially with smaller expert configurations. Although response time gradually increases with a higher number of experts, it remains considerably lower than the base model in all cases. These results highlight the efficiency benefits of MoE models in achieving faster response generation while supporting improved performance across domains.

F. Overall Gain Analysis

Figure 5 illustrates the overall gain G (as defined in Eq. (1)) achieved by the MoE models across varying sample sizes, using 8 experts for both BERT and DeBERTa models. The gain metric balances prediction quality with analysis and response latency. As expected, small sample sizes lead to lower analysis time but poorer prediction performance, while very large sample sizes incur higher latency with diminishing returns. Across both datasets, the trend is consistent, with MedMCQA showing marginally higher gain values—likely due to the dataset's more domain-focused structure. Notably, the optimal gain is observed at a moderate sample size around 400, suggesting a practical trade-off point for real-time applications.

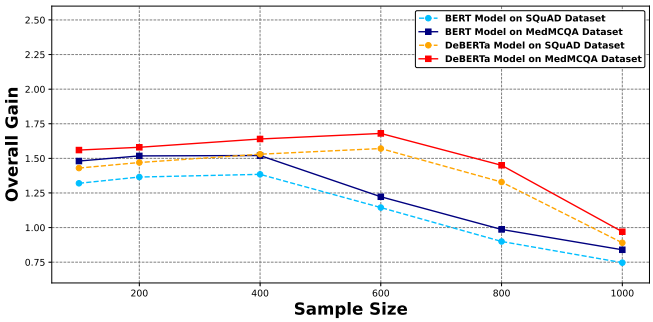

Fig. 5: Overall gain as a function of sample size. Both used 8 experts.

Analyzing all experimental results collectively, our evaluations confirm the effectiveness and generalizability of the MMEC framework. The MoE models consistently outperform their base counterparts (BERT and DeBERTa) across various metrics, including F1 score, response generation time, and overall gain. Moderate configurations (typically between 6 and 8 experts) achieve the best balance between accuracy and efficiency, with diminishing returns observed at higher expert counts. These trends hold across both general-purpose (SQuAD V2.0) and domain-specific (MedMCQA) datasets,

TABLE IV: Response Generation Time for different numbers of experts, n (0 means base), unit in seconds

Dataset	Model	n=0	n=1	n=2	n=4	n=8	n=16	n=32
SQuAD	BERT	59.45	4.43	4.47	4.54	4.95	5.29	7.09
MedMCQA	BERT	37.48	4.35	4.48	4.57	4.81	5.30	6.88
SQuAD	DeBERTa	45.39	3.59	3.62	3.77	3.88	3.93	4.18
MedMCQA	DeBERTa	41.88	3.02	3.17	3.26	3.95	3.28	3.49

and are validated by both BERTScore and SQuAD evaluation metrics. Furthermore, the response time reductions and sample-efficient task analysis contribute to a more practical deployment model for edge devices. The gain metric further reinforces that our expert selection strategy, combined with lightweight analysis, yields robust performance improvements under constrained conditions. These findings collectively demonstrate that MMEC can intelligently and efficiently support dynamic MoE deployment on mobile edge platforms.

V. Concluding Remarks and Future Work

In this work, we have proposed MMEC, a framework for efficient deployment of MoE models on mobile edge devices. Our approach leverages a lightweight task analyzer to identify the most suitable model configuration based on a small number of sampled queries. Through experiments on SQuAD V2.0 and MedMCQA datasets using both BERT and DeBERTa-based models, we demonstrated that the MoE architecture consistently improves performance across multiple metrics, including F1 score, response generation time, and overall gain. The results show that models with 6 to 8 experts strike the best balance between performance and efficiency.

Using the comprehensive gain metric, we identified practical sample sizes for the analyzer to make informed decisions without incurring high latency. These findings show that MMEC can effectively support intelligent expert selection under real-world resource constraints.

Our future work will explore refining the gating mechanism and implementing dynamic expert selection strategies to further reduce resource usage. These can improve MMEC's applicability in real-time and resource-constrained environments.

References

[1] S. Pavlitska, C. Hubschneider, L. Struppek, and J. M. Zöllner, "Sparsely-gated mixture-of-expert layers for cnn interpretability," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023, pp. 1–10.

[2] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-trained language models and their applications," *Engineering*, vol. 25, pp. 51–65, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809922006324

[3] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1810.09291*, 2018. [Online]. Available: https://arxiv.org/abs/1810.09291

[4] S. Sadiq and S. R. M. Zeebaree, "Distributed systems for machine learning in cloud computing: A review of scalable and efficient training and inference," *ijcs*, vol. 13, no. 2, Apr. 2024.

[5] O. Friha, M. Amine Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, and N. Ghoualmi-Zine, "Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 5799–5856, 2024.

[6] M. Lewis, A. v. d. Oord, V. Sze, C. Riquelme, M. Bosma, Y. Bachrach, N. de Freitas, G. Hinton, and J. Dean, "Sparsely-gated mixture-of-experts for efficient deep learning," *arXiv preprint arXiv:2308.15030*, 2023. [Online]. Available: https://arxiv.org/pdf/2308.15030

[7] C. Liu and J. Zhao, "Resource allocation for stable llm training in mobile edge computing," in *Proceedings of the Twenty-Fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MOBIHOC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 81–90. [Online]. Available: https://doi.org/10.1145/3641512.3686358

[8] P. F. Brown, S. A. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992. [Online]. Available: https://www.aclweb.org/anthology/J92-4003/

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://www.bioinf.jku.at/publications/older/2604.pdf

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019. [Online]. Available: https://aclanthology.org/N19-1423/

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI Blog*, 2018.

[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020. [Online]. Available: https://jmlr.org/papers/volume21/20-074/20074.pdf

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991. [Online]. Available: https://doi.org/10.1162/neco.1991.3.1.79

[15] X. Zhang, Y. Wang, Y. Zhao, J. Chen, W. Xu, and M. Zhang, "Edgeshard: Efficient llm inference via collaborative edge computing," *arXiv preprint arXiv:2405.14371*, 2024. [Online]. Available: https://arxiv.org/abs/2405.14371

[16] P. Rajpurkar, R. Jia, and P. Liang, "Squad explorer," 2016, accessed: 2025-01-23. [Online]. Available: https://rajpurkar.github.io/SQuAD-explorer/

[17] A. Pal, L. K. Umapathi, and M. Sankarasubbu, "Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering," in *Proceedings of the Conference on Health, Inference, and Learning*, ser. Proceedings of Machine Learning Research, G. Flores, G. H. Chen, T. Pollard, J. C. Ho, and T. Naumann, Eds., vol. 174. PMLR, 07–08 Apr 2022, pp. 248–260. [Online]. Available: https://proceedings.mlr.press/v174/pal22a.html

[18] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr