Research on Top-K MMoE and Load Balancing

Yuntao Zheng
yuntaozh@andrew.cmu.edu

Jiachun Xu
jiachunx@andrew.cmu.edu

Yichen Lu
yichenl5@andrew.cmu.edu

Abstract

MMoE[3] is one of state-of-the-art approaches in multi-task learning, demonstrating strong performance and improved trainability even when the Pearson correlation between tasks is low. A typical strategy to further enhance MMoE's performance is to increase the number of experts, which often expands model capacity and improves results. However, this also increases computational demands. Sparsely-Gated MoE[5] addresses this by introducing Top-K MoE, which reduces computation to only K experts while preserving a larger effective model capacity. Another challenge with a high number of experts is load imbalance—a phenomenon where the gating network consistently assigns large weights to only a few experts, leading to underfitting of others. To mitigate this, a common industrial approach is to add a dropout layer to each gate, while Sparsely-Gated MoE and OLMoE[4] explicitly introduce load balancing losses. In this study, we explore the Top-K MMoE and evaluate the effectiveness of these two load balancing strategies. We show that on the UCI Census-Income (KDD) dataset, load balancing loss performs much better than dropout while router-z loss contributes little to the stability and quality of the MMoE model.

1 Introduction

For recommendation systems, Multi-gate Mixture-of-Experts, i.e. MMoE[3], is a widely-used model in industry. It shows strong advantages when tasks are less related and some trainability benefit while adapting to the inherent randomness in training data and model initialization. However, since MMoE is based on MoE model, polarization still acts as a problem for model training, where most of the load is routed to some experts while other experts receive less updates.

In this work, we propose a Top-K MMoE model, stacking sparse experts with a multi-gated network. We also implement several load-balancing losses to compare their contribution to routing load and utilize router-z loss to stabilize the training process.

For simplicity, in the following sections, we will use l_{LB1} to refer to the load balancing loss employed in the Sparsely-Gated MoE, l_{LB2} to represent the load balancing loss introduced by OLMoE, and l_{RZ} to denote the router-z-loss from OLMoE[4].

2 Related Works

Mixture of Experts, i.e. MoE[2], is an efficient neural network model, making its sub-networks, i.e. experts, focus on different parts of the task. Based on MoE, Multi-gated Mixture of Experts, i.e. MMoE[3], uses a gating network to dynamically assign experts' importance to the target value. However, training MMoE typically incurs higher computational costs. Sparsely-Gated MoE[5]

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

proposes a sparse structure to increase model capacity while keeping the training parameters in a relatively small scale. Moreover, during model training, polarization often arises as a significant issue in the absence of proper constraints on load routing. To alleviate the polarization for MMoE training, YouTube[6] implements the dropout layer, randomly closing gates for some experts, which contributes to more balanced training load. To impose a soft constraint, Sparsely-Gated MoE[5] introduces a load balancing loss to give penalty to harsh polarization. OLMoE[4] simplifies the load balancing loss function based on Sparsely-Gated MoE to reduce computational overhead while maintaining efficient expert utilization.

3 Top-K MMoE

In this work, we propose Top-K MMoE, a novel approach that builds upon the MoE[2] framework and the Multi-gate MoE model. By introducing a top-K selection mechanism, Top-K MMoE enhances the capabilities of MMoE[3] while maintaining computational efficiency. This method aims to optimize the allocation of computational resources and improve task-specific performance without increasing the overall floating-point operations.

3.1 Model Architecture

The architecture of Top-K MMoE is illustrated in Figure 1. The model consists of a shared pool of experts, where each expert is designed to specialize in different aspects of the input data. For each task, a dedicated gate selects the top-K experts from the pool based on their relevance to the task. This selection is performed using a gating mechanism that computes scores for each expert and dynamically activates the top-K experts for a given input.

The proposed top-K selection mechanism allows Top-K MMoE to focus computational efforts on the most relevant experts for each task, reducing redundancy and improving the efficiency of the model. This approach not only enhances the load balancing across experts but also leads to better utilization of the expert network, ultimately improving task-specific performance. Furthermore, the architecture ensures that the overall floating-point computation remains constrained, making it a scalable and efficient solution for multi-task learning.

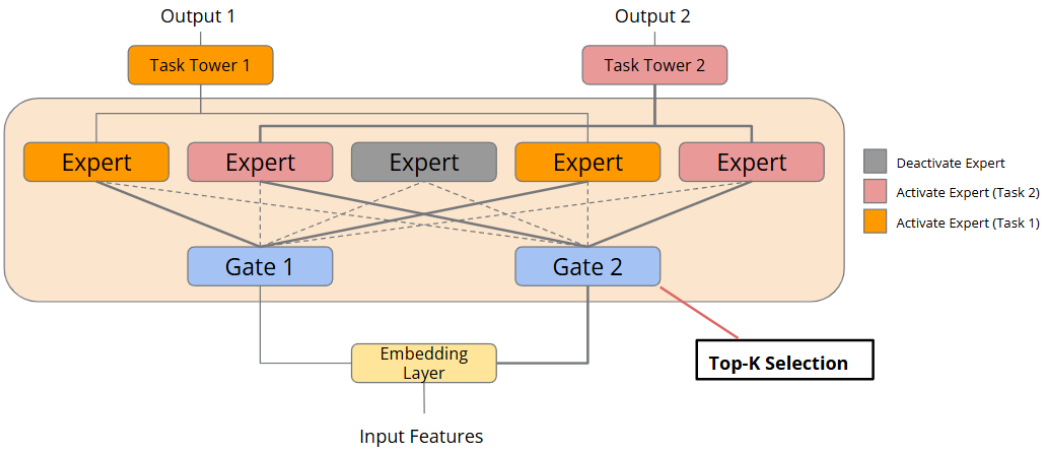

Figure 1: The architecture of the proposed Top-K MMoE model. We choose $K = 2$ and $\mathcal{M} = 5$ as the example(we select Top 2 experts from 5 total experts for each task).

3.2 Polarization in MoE Architecture

During training, the MoE[2] architecture may encounter polarization issues, where one expert becomes highly optimized while others remain underfit. This leads the gating mechanism to increasingly favor the well-trained expert, resulting in significant imbalances in expert utilization. To address this challenge, we explore four different methods to mitigate the polarization problem from previous

works. The effectiveness of these approaches is evaluated through experiments, which are detailed in Section 4. Below, we provide an overview of the methods investigated:

Dropout[6]: Dropout is introduced at the expert level during training to randomly deactivate some experts. This forces the model to diversify its utilization of experts, thereby reducing reliance on any single expert and encouraging more balanced training. No specific loss function is introduced in this method, as the balancing effect is achieved through random deactivation.

Load Balancing Loss (Sparsely-Gated MoE)[5]: The load balancing loss from Sparse MoE encourages even distribution of load across experts by optimizing the following formula:

$$L_{LB} = w_{\text{importance}} \cdot CV(\text{Importance}(X))^2 + w_{\text{load}} \cdot CV(\text{Load}(X))^2$$

where

$$\text{Importance}(X) = \sum_{x \in X} G(x), \quad \text{Load}(X) = \sum_{x \in X} P(x, i),$$

and $CV(\cdot)$ is the coefficient of variation. The loss penalizes imbalances in both importance and load across experts.

Load Balancing Loss (From OLMOE)[4]: The OLMOE load balancing loss builds on Sparse MoE's concept by directly combining importance and load into a single loss function:

$$L_{LB} = N_E \cdot \sum_{i=1}^{N_E} f_i \cdot P_i,$$

where N_E is the number of experts, f_i is the activation fraction of expert i, and P_i represents the expert's load. This loss explicitly optimizes for balanced expert utilization.

Router-Z Loss (From OLMOE)[4]: Router-Z loss introduces entropy regularization to further balance the expert selection probabilities:

$$L_{RZ}(x) = \frac{1}{B} \sum_{i=1}^{B} \log \left(\sum_{j=1}^{N_E} \exp(x_j^{(i)}) \right)^2,$$

where B is the batch size, $x_j^{(i)}$ is the gating score of expert j for input i, and N_E is the total number of experts. By penalizing overly sparse or dense activations, this loss achieves a better trade-off between efficiency and utilization.

These methods collectively address the polarization problem from multiple perspectives, providing a comprehensive solution for improving expert utilization in the MoE framework.

4 Experiment

4.1 Dataset

To reproduce the results in MMoE [3], we use the same dataset as that in the original paper, i.e., the UCI Census-Income (KDD) dataset [1]. This dataset is extracted from the 1994 census database and contains 299,285 instances of demographic information of American adults.

We replicate the setups used in the original work and construct two multi-task learning problems from this dataset by setting some of the features as prediction targets. Additionally, we calculate the absolute value of the Pearson correlation of the task labels:

- **Problem 1:**
  - **Task 1:** Predict whether the income exceeds \$50K.
  - **Task 2:** Predict whether this person's marital status is "never married".
  - **Absolute Pearson correlation:** 0.1772
- **Problem 2:**
  - **Task 1:** Predict whether the education level is at least college.
  - **Task 2:** Predict whether this person's marital status is "never married".
  - **Absolute Pearson correlation:** 0.1666

4.2 Baseline and Our Methods

We use MMoE[3] as the baseline and implement our proposed Top-K MMoE in five variations to evaluate the impact of additional loss functions and dropout applied to the gating mechanism. The results can be found in Table 1.

Models	Activated/Total Params	Problem 1(AUC)	Problem 2(AUC)
MMoE	11978/11978	0.9353/0.9877	0.9668/0.9907
Top-K MMoE	12898/45586	0.9379/0.9897	0.9780/0.9907
Top-K MMoE + Dropout	12898/45586	0.9389/0.9899	0.9751/0.9910
Top-K MMoE + l_{LB1}	12898/45586	**0.9391/0.9916**	**0.9755/0.9923**
Top-K MMoE + l_{LB2}	12898/45586	0.9304/0.9902	0.9735/0.9915
Top-K MMoE + l_{LB2} + l_{RZ}	12898/45586	0.9334/0.9898	0.9735/0.9915

Table 1: Comparison between baseline approach and our proposed methods. The table presents the number of activated parameters, total parameters, and performance on two tasks. In the performance column, each cell displays two values corresponding to the AUC scores for the respective tasks. For hyperparameters, we set the dropout rate to 0.2, the weight of l_{LB1} to 0.01, the weight of l_{LB2} to 1×10^{-4}, and the weight of l_{RZ} to 1×10^{-3}.

4.3 Effect of Dropout

To alleviate the polarization phenomenon in Top-K MMoE—where a small number of experts dominate the majority of the weights while the others have weights close to zero—we first examine the impact of adding a dropout layer to the gates, a technique commonly used in industry. The tables below present the performance results for various dropout rates.

Gate Dropout	**Task1 AUC**	**Task2 AUC**	**Weight Variance**	**#Ineffective Expert**
0	0.9379	0.9897	0.032/0.019	14/12
0.2	0.9389	0.9899	0.029/0.035	12/12
0.4	0.9370	0.9916	0.038/0.036	12/10
0.6	0.9363	0.9909	0.038/0.039	11/10
0.8	0.9273	0.9911	0.030/0.037	11/9

Table 2: Performance of Top-K MMoE on Problem 1 with varying dropout rates applied to the gating mechanism. Here, *weight variance* refers to the variance of the average weights across different experts, and *#ineffective experts* represents the number of experts whose average weights are close to zero. For the hyperparameters, the model uses a total of 16 experts with $k = 4$.

Gate Dropout	**Task1 AUC**	**Task2 AUC**	**Weight Variance**	**#Ineffective Expert**
0	0.9780	0.9907	0.039/0.022	13/13
0.2	0.9750	0.9910	0.040/0.046	13/13
0.4	0.9694	0.9911	0.040/0.038	12/11
0.6	0.9672	0.9916	0.044/0.042	12/10
0.8	0.9603	0.9897	0.029/0.039	12/9

Table 3: Peformance of Top-K MMoE on Problem 2 using different dropout rate in gates.

From the table 2 and table 3, we can conclude that

1. Dropout fundamentally differs from the load balancing loss, as it does not aim to balance the load across experts, and therefore cannot effectively reduce weight variance.
2. The impact of dropout is limited; as the gate dropout rate increases, the number of ineffective experts decreases only slightly.
3. When the dropout rate increases from 0.6 to 0.8, Task 1's performance drops significantly, indicating that an excessively high dropout rate can negatively affect the performance of multi-task learning.

4. Optimal performance is observed at a gate dropout rate of 0.2, with approximately four effective experts. This suggests that the dropout mechanism may perform well when the number of experts is small but is less effective for models like Top-K MMoE that have a larger number of experts.

4.4 Effect of l_{LB1}

To tackle the polarization phenomenon more effectively, we implement the load balancing loss to penalize the model if it routes most tokens to few expert models. We combine the original task loss with load balancing loss to force the model to decrease the imbalance and routes tokens in a fair way. The tables below represent results with different loss weights.

Loss Weight	Task1 AUC	Task2 AUC	Weight Variance	#Ineffective Expert
0	0.9381	0.9890	0.0005/0.0012	2/4
0.001	0.9348	0.9899	0.0061/0.0083	9/11
0.005	0.9375	0.9907	0.0008/0.0016	5/5
0.01	0.9355	0.9899	0.0009/0.0011	3/4
0.05	**0.9363**	**0.9908**	**0.0001/0.0001**	**0/0**
0.1	0.9307	0.9899	0.0001/0.0001	0/0

Table 4: Performance of Top-K MMoE on Problem 1 with different load balancing loss. Here, *weight variance* and *#ineffective experts* follow the definition in Table 1. For the hyperparameters, the model uses a total of 16 experts with $k = 4$.

Loss Weight	Task1 AUC	Task2 AUC	Weight Variance	#Ineffective Expert
0	0.9721	0.9906	0.0506/0.0196	14/14
0.001	0.9736	0.9910	0.0026/0.0092	6/11
0.005	0.9758	0.9899	0.0012/0.0017	5/5
0.01	0.9755	0.9923	0.0010/0.0010	3/3
0.05	0.9738	0.9911	0.0001/0.0001	1/1
0.1	**0.9755**	**0.9913**	**0.0001/0.0001**	**1/1**
0.2	0.9756	0.9913	0.0001/0.0001	1/2
0.5	0.9690	0.9908	0.0001/0.0001	1/1

Table 5: Performance of Top-K MMoE on Problem 2 with different load balancing loss.

From the table 4 and 5, we can conclude that:

1. Since the load balancing loss is combined with the original objective function, the model will be guided by the load balancing loss to alleviate the polarization phenomenon.
2. Compared to dropout technique, the load balancing loss is more effectively for a large number of experts, as it can result in nearly no ineffective experts by tuning the loss weight.
3. Optimal performance for the two problems is observed at a loss weight of 0.05-0.1, with nearly no ineffective experts.

4.5 Effect of l_{LB2}

The results in Table 6 and Table 7 demonstrate the effectiveness of l_{LB2} on Problem 1 and Problem 2, respectively. Compared to the baseline, the model trained with OLMO[4] balancing loss shows significant improvements in weight variance, suggesting that the model achieves a better load balance among experts. This reduction in weight variance corresponds to fewer ineffective experts and indicates that the model utilizes its expert resources more effectively. Additionally, this improvement in load balancing contributes to performance gains on both problems, as evidenced by higher AUC scores on both tasks. The results affirm the importance of load balancing in improving the overall efficiency and performance of the model.

OLMO Coef	Task1 AUC	Task2 AUC	Weight Variance	Ineffective Expert
0	0.9371	0.9899	0.0333/0.0225	11/10
0.01	0.9251	0.9893	0.0103/0.0095	9/8
0.05	0.9201	0.9896	0.0107/0.0108	11/9
0.001	0.9259	0.9901	0.0075/0.0074	9/6
0.0001	**0.9304**	**0.9902**	**0.0024/0.0010**	**1/0**
0.00001	0.9364	0.9912	0.0021/0.0039	2/4

Table 6: Performance of Top-K MMOE on **Problem 1** with different OLMO coefficient.

OLMO Coef	Task1 AUC	Task2 AUC	Weight Variance	Ineffective Expert
0	0.9371	0.9899	0.0333/0.0225	11/10
0.01	0.9529	0.9904	0.0114/0.0110	9/9
0.05	0.9425	0.9897	0.0109/0.0097	10/10
0.001	0.9371	0.9904	0.0089/0.0061	8/6
0.0001	**0.9735**	**0.9915**	**0.0014/0.0021**	**0/0**
0.00001	0.9764	0.9926	0.0039/0.0032	4/2

Table 7: Performance of Top-K MMOE on **Problem 2** with different OLMO coefficient.

4.6 Effect of l_{RZ}

The impact of l_{RZ} is summarized in Table 8 and Table 9 for Problem 1 and Problem 2, respectively. Introducing the Router-Z loss has a pronounced effect on reducing weight variance, leading to a more balanced distribution of task-specific loads across the experts. This reduction in weight variance is accompanied by a consistent decrease in the number of ineffective experts, indicating improved utilization of expert capacity. While the improvements in AUC scores for Task 1 and Task 2 are more marginal compared to l_{LB2}, the results suggest that l_{RZ} provides an effective complementary approach to enhancing model efficiency by reducing redundancy in expert activation. These findings highlight the potential of Router-Z loss to refine the model's performance by optimizing task allocation.

Router-z Coef	Task1 AUC	Task2 AUC	Weight Variance	Ineffective Expert
0.0	0.9304	0.9902	0.0024/0.0010	1/0
1e-5	0.9334	0.9898	0.0026/0.0033	2/1
1e-4	0.9287	0.9906	0.0018/0.0007	1/0
1e-3	0.9355	0.9897	0.0021/0.0028	1/1

Table 8: Effectiveness of Router-Z loss on Problem 1

Router-z Coef	Task1 AUC	Task2 AUC	Weight Variance	Ineffective Expert
0.0	0.9735	0.9915	0.0014/0.0021	0/0
1e-5	0.9744	0.9899	0.0017/0.0025	0/0
1e-4	0.9726	0.9904	0.0029/0.0017	3/0
1e-3	0.9700	0.9907	0.0030/0.0018	3/1

Table 9: Effectiveness of Router-Z loss on Problem 2

5 Conclusion

We propose Top-K MMoE model, which enhances computational efficiency through dynamic top-K expert selection. Our experiments demonstrate that this approach improves resource utilization and task-specific performance. Additionally, we evaluated four methods—Dropout, Sparse MoE Load Balancing Loss, OLMOE Load Balancing Loss, and Router-Z Loss—to mitigate polarization. These methods effectively reduced weight variance, balanced expert utilization, and minimized ineffective

experts, with OLMOE and Router-Z Loss showing particularly strong results. Our findings highlight the importance of balancing expert load to enhance the efficiency and performance of MoE models. Future work will explore adaptive gating mechanisms and broader applications to further refine and validate these approaches.

References

[1] Arthur Asuncion and David Newman. *UCI machine learning repository*, 2007.

[2] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991.

[3] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 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*, 2018.

[4] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

[5] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[6] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed H. Chi. Recommending what video to watch next: a multitask ranking system. *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019.