Weekly report (JAN 14, 2025 - JAN 20, 2025)

Stanley Zheng

January 21, 2025

Abstract

In the last week, I mainly read [2], [4], [6], [7], [9], [8], and [1]. In these papers, I carefully read [3] and [9] and just roughly read the others. Also, I watched [10], [14], [13], [11], and [12]. In this report, I'll do a summary of the papers and weite a brief future plan.

1 Improving MoE models

In this week, all the papers I read is about improving MoE models. These papers can be roughly divided into three categories:

1. Improving the MoE models with more efficiency: This includes [7], [1], [4], [9], and [8]. You can find words like "efficient", "limited GPU memory", "low-latency" ... I just summarize the main idea of these papers is making MoE models more efficient.
2. Improving the MoE models with better performance: This includes [2].
3. Other types of improvements: This includes [6].

1.1 Improving the MoE models with more efficiency

All the papers here actually share a same story: DNNs or Transformer models are getting larger and larger due to the scalling laws, however, the current computation resources are limited. So, for many institutions, they do not have enough GPUs to train a very big model.

Then the MoE models come to the stage. Since we only use part of the parameters in the model, so the computation is affordable. However, the memory cost is still high. Also, for MoE models, they naturally need more GPU memory to achieve the same performance as the dense models.

This lead to the question: How can we make the MoE models more efficient? I conclude that there are two ways to improve the efficiency of MoE models:

1. Model-Level: This includes [7] and [1].
2. Caching-Level(Offloading-Efficiency): This includes [4], [9] and [8].

1.1.1 Model-Level

The model-level is just means that the researchers mainly do things on the MoE models, like changing structures and so on.

[1] is very strightforward, it just shows that each MoE layer can only activate one expert with no significant loss of performance. They just cut off the most insignificant expert one-by-one, and in the end every MoE layer only have one expert. With less experts, the inference speed just go up.

[7] is more complex. They contains two steps: 1. They find a way to merge the important experts into one expert. 2. They find they can compress the result expert due to the low dimensionality.

1.1.2 Caching-Level(Offloading-Efficiency)

The main idea here is that we try to utilize the memory of the CPU. So we only put the experts we need into the GPU memory, and if we do it in this way, we'll face high latency since we are loading and Offloading parameters from CPU and GPU all the time. However, if we can know what experts we'll need in the next layer, this process can be very fast. [4], [9] and [8] just uses different way to predict the experts we need in the future.

[4] uses "the information in the correlation tables" to predict. [9] uses EAMs and EAMC to record the information, and then calculat the similarity to predict. It's actually also about "correlation". When I read [9], I actually thought that we are just do some prediction of the experts in the future layers, why don't we just use the neural network to predict. We all know that current neural network can find relations people can not find, and they perform very well in simple datasets. This is actually what [8] do. They propose a learned predictor, which is a neural network.

1.2 Improving the MoE models with better performance

Actually, maybe this paper can also be classified into the first category, but this paper is like a technique report and it do not explain very well in the Abstract and Introduction part. Also, it do not have the Conclusion. I literally do not know what they actually do after reading the first two part. Still, they claim many times that their model have higher throughput and lower latency. I think maybe they do not care about the efficiency too much, instead they care about a comprehensive performance.

1.3 Other types of improvements

[6] is quite different, it's about help researchers to implement parallelization more easily. This just remind me of the [5]. [5] shows a very complex model partitioning technique, which inspired many later works.

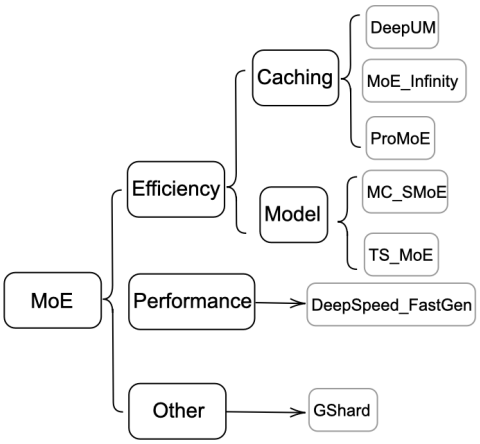

Figure 1: Mindmap Structure

1.4 Small Summary

I draw a mindmap structure figure which present in Figure 1. I actually do not have too much things to say here, but I wonder maybe we could combine the two ways to improve the efficiency of MoE models.

Also, I want to talk about the writing, since the style is so classical, just like what I learn in academic writing class. You just write the current works' limitation, and introduce current solutions(if there are solutions), then you just introduce your improvements on these solutions. Taking [9] as an example, I think the writing style is pretty standard. I don't need to know too much about the background still I can understand the main idea of the paper. The whole paper is actually a detailed explanation of the Introduction part. So, it's pretty easy to follow.

2 Future Plan

My future plan is quite simple. I'll just continue to read these papers since only [9], I read the whole passage and for others I just read the Abstract, Introduction and Conclusion.

I'll also continue to watch the videos serier of Li, I think his videos is a quite simple and helpful way for me to understand more in AI field. Though these papers are not about MLsys at all.

I will also try to learn how to use the Hugging Face, I actually already started but I just dorp it for a while, I'll try to finish the tutorial they provided.

References

[1] Tianyu Chen et al. *Task-Specific Expert Pruning for Sparse Mixture-of-Experts*. Accessed: 2024-09-09. 2022. URL: https://doi.org/10.48550/arXiv.2206.00277.

[2] Connor Holmes et al. *DeepSpeed-FastGen: High-throughput Text Generation for LLMs via MII and DeepSpeed-Inference*. Accessed: 2024-09-09. 2024. URL: https://doi.org/10.48550/arXiv.2401.08671.

[3] *How to Read a Paper*. Accessed: 2024-09-09. URL: https://web.stanford.edu/class/ee384m/Handouts/HowtoReadPaper.pdf.

[4] Jaehoon Jung, Jinpyo Kim, and Jaejin Lee. "DeepUM: Tensor Migration and Prefetching in Unified Memory". In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. Vancouver BC Canada: ACM, Jan. 2023, pp. 207–221. URL: https://doi.org/10.1145/3575693.3575736.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60 (May 2012), pp. 84–90.

[6] Dmitry Lepikhin et al. *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*. Accessed: 2024-09-09. 2020. URL: https://doi.org/10.48550/arXiv.2006.16668.

[7] Pingzhi Li et al. *Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy*. Accessed: 2024-09-09. 2024. URL: https://doi.org/10.48550/arXiv.2310.01334.

[8] Xiaoniu Song, Zihang Zhong, and Rong Chen. *ProMoE: Fast MoE-based LLM Serving using Proactive Caching*. Accessed: 2024-09-09. 2024. URL: https://doi.org/10.48550/arXiv.2410.22134.

[9] Leyang Xue et al. *MoE-Infinity: Offloading-Efficient MoE Model Serving*. Accessed: 2024-09-09. 2024. URL: http://arxiv.org/abs/2401.14361.

[10] 算法猪立业. 认识混合专家模型（*MoE*）. Accessed: 2025-1-17. 2024. URL: https://www.bilibili.com/video/BV1Ep421m7cx.

[11] 跟李沐学AI. *GPT, GPT-2, GPT-3* 论文精读【论文精读】. Accessed: 2025-1-17. 2022. URL: https://www.bilibili.com/video/BV1AF411b7xQ.

[12] 跟李沐学AI. *OpenAI Codex* 论文精读【论文精读】. Accessed: 2025-1-18. 2022. URL: https://www.bilibili.com/video/BV1iY41137Zi.

[13] 跟李沐学AI. 如何读论文【论文精读·*1*】. Accessed: 2025-1-15. 2021. URL: https://www.bilibili.com/video/BV1H44y1t75x.

[14] bryanyzhu 跟李沐学AI. 双流网络论文逐段精读【论文精读】. Accessed: 2025-1-17. 2022. URL: https://www.bilibili.com/video/BV1mq4y1x7RU.