CS498 Project - Enable Expert Parallelism for Universal Checkpointing

Xinyu Lian
lian7@illinois.edu

Hao Chen
chen3@illinois.edu

Tengjun Jin
tengjun2@illinois.edu

1 Introduction

The Mixture-of-Experts (MoE) paradigm has become a cornerstone for scaling large neural networks by enabling conditional computation: only a small subset of specialized "expert" subnetworks is activated per input, dramatically increasing model capacity without a proportional rise in computational cost. Modern MoE-based language models—ranging from Google's Switch Transformer and GLaM to open-source efforts like Mixtral and DeepSeek—routinely push parameter counts into the trillions while keeping inference and training costs manageable. A key enabler of this scalability is expert parallelism: distributing different experts across devices so that multiple experts can be computed in parallel.

Despite its benefits, expert parallelism introduces significant operational complexity. Checkpoint files become tightly coupled to a specific parallelism strategy, since model and optimizer states are sharded differently depending on the number of experts and devices. In practice, resuming training with a different GPU count or expert configuration requires custom conversion scripts—or is simply unsupported—hindering flexibility in both research exploration and production deployment. Existing checkpointing frameworks (e.g., PyTorch DDP, ZeRO, and Megatron's 3D parallelism support) offer limited reconfiguration capabilities, and recent universal checkpointing approaches focus primarily on dense models, leaving MoE architectures unaddressed.

In this work, we present **UCP-EP**, an extension of the Universal Checkpointing framework in DeepSpeed that enables seamless reconfiguration of expert parallelism in MoE models. By introducing a new "Unique-MoE" pattern for matching and transforming sharded expert parameters, along with two mapping operators (Local-to-Global and Global-to-Local ID translation), UCP-EP decouples checkpoint formats from any specific expert layout. We demonstrate that UCP-EP can (1) decrease or increase the expert parallelism degree mid-training, (2) transition between expert and data parallelism, and (3) preserve training fidelity across all configurations, all without manual checkpoint conversion scripts. Our contributions pave the way for more flexible and efficient experimentation with large-scale MoE architectures.

2 Background

2.1 MoE Model

The Mixture-of-Experts (MoE) paradigm, first formalized in Jacobs et al. [1991], has emerged as a powerful framework for scaling neural networks by decomposing a large model into a collection of smaller, specialized "expert" subnetworks, each of which focuses on a different region of the input space. In practice, an MoE layer consists of N expert networks (e.g. small feed-forward blocks), together with a trainable gating network that, for each input token or example, computes a sparse probability distribution over the experts and routes the input only to the top-k experts, as shown in Figure 1. This sparsity makes it possible to dramatically increase the total parameter count (and thus representational capacity) without incurring a proportional increase in computational cost, since only a small subset of experts is activated per token.

Preprint. Under review.

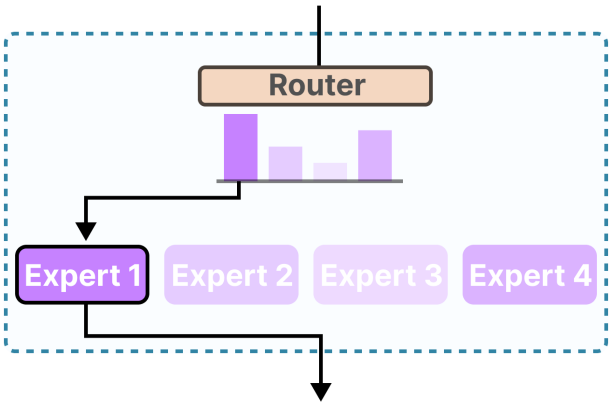

Figure 1: A Visual Guide to Mixture of Experts (MoE).

The original Adaptive Mixtures of Local Experts model Jacobs et al. [1991] demonstrated that, by dividing the learning problem into subtasks—each handled by an expert—and coordinating them via a gating mechanism, one can achieve improved data efficiency and generalization. Since then, MoE architectures have been refined with techniques such as load-balancing losses to prevent expert collapse, auxiliary routing regularizers to smooth expert utilization, and hierarchical or multi-stage gating to capture more nuanced patterns Jiang et al. [2024a], DeepSeek-AI et al. [2024].

Today, Mixture-of-Experts has become a de facto trend in large-scale model releases: from Google's Switch Transformer and GLaM to recent open-source innovations like Mixtral, DeepSeek, Llama4 and several of the latest GPT-style variants, MoE layers are being incorporated to push parameter counts into the trillions while keeping inference costs manageable. This surge reflects the community's recognition that conditional computation via dynamic expert selection is one of the most effective—and economical—ways to continue scaling deep learning models without linear increases in FLOPs.

2.2 Expert Parallelism

The MoE architecture has emerged as a pivotal approach to scaling deep learning models efficiently. By activating only a subset of specialized "experts" for a given input, MoE models achieve significant computational savings while maintaining high performance. Expert parallelism, wherein different experts are distributed across multiple devices, is central to this efficiency, enabling large-scale models to be trained and inferred effectively.

Figure 2 illustrates the concept of expert parallelism in Mixture-of-Experts (MoE) models. On the left, all experts (Expert 1, Expert 2, Expert 3) are colocated on a single GPU, and the router sends each input token to the appropriate expert within that device. On the right, expert parallelism is applied: each expert is placed on a separate GPU (GPU 1, GPU 2, GPU 3), and the router distributes tokens across GPUs according to the expert assignment. This enables parallel computation of different experts and requires inter-GPU communication for token routing. By distributing experts across multiple devices, expert parallelism allows MoE models to scale to larger sizes and higher throughput, while maintaining efficient utilization of hardware resources.

Early implementations such as GShard [Lepikhin et al., 2020] and the Switch Transformer [Fedus et al., 2022] laid the foundation by demonstrating the viability of sparse activation and expert distribution. GShard utilized a top-2 gating mechanism with all-to-all communication to distribute computations across devices, while Switch Transformer simplified this by employing a top-1 gating strategy, reducing communication overhead.

Recent research has focused on addressing the challenges associated with expert parallelism, particularly communication bottlenecks and load imbalance.[Cai et al., 2024] introduced the Shortcut-connected Mixture-of-Experts (ScMoE) architecture, which decouples communication from com-

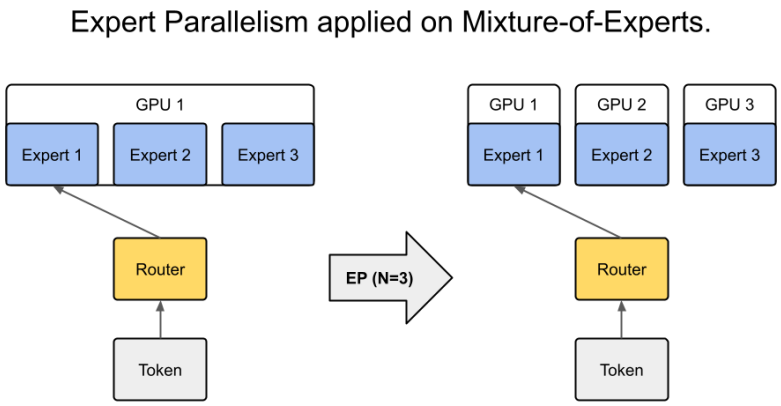

Figure 2: Visualization of Expert Parallelism from NVIDIA NeMo. Experts are distributed across GPUs with parallel computation and inter-GPU token routing.

putation, allowing for substantial overlap between the two. This approach achieved training speed improvements of up to 30% and inference speedups of 40% in specific hardware environments.

Yao et al. [2024] proposed *ExFlow*, an optimization technique that leverages inter-layer expert affinity to reduce cross-GPU communication during inference. By aligning expert placement with token routing patterns, ExFlow achieved up to a 2.2× improvement in inference throughput without requiring model fine-tuning.

To tackle load imbalance, *MoETuner* [Go and Mahajan, 2025] introduced an Integer Linear Programming (ILP) formulation to optimize expert-to-GPU assignments. This method minimized inter-GPU token routing costs and balanced token processing loads, resulting in end-to-end speedups of up to 17.5% in multi-node inference scenarios.

Further advancements include *Lancet* [Jiang et al., 2024b], which employs whole-graph computation-communication overlapping to accelerate MoE training. By overlapping non-MoE computations with communication phases, Lancet reduced non-overlapping communication time by up to 77% and achieved a 1.3× speedup over prior solutions.

A recent survey by [Cai et al., 2025] provides a comprehensive taxonomy of MoE models, detailing algorithmic and system-level designs, open-source implementations, and empirical evaluations. This serves as a valuable reference for future research.

In summary, expert parallelism in MoE models has evolved significantly, with recent research addressing key challenges in communication efficiency and load balancing. These advancements have enhanced the scalability and performance of large-scale deep learning models, solidifying expert parallelism as a critical component in modern AI architectures.

3 Challenges

The distributed checkpoints are highly coupled to specific model parallelism strategy. For example, data parallelism (e.g., PyTorch DistributedDataParallel Li et al. [2020]), one of the most commonly used parallelism strategies, replicates model weights and optimizer states across GPUs, and therefore only one rank (e.g., rank 0) saves the entire model states in a checkpoint file. When resuming training, each data parallel worker loads the same checkpoint file before launching the training iterations. Other parallelism techniques, such as ZeRO Rajbhandari et al. [2019] and 3D parallelism Narayanan et al. [2021], shard model parameters and optimizer states across GPUs. Since the model states are partitioned across GPUs, each GPU separately saves and loads checkpoint files that contain only a fraction of model state it owns.

Model parallelism-coupled distributed checkpoints lead to a relatively simple checkpointing saving/loading pipeline, because the checkpointing loading is simply the reverse of checkpoint saving process for each worker. This design also carries a performance advantage because distributed checkpoint saving requires to block all parallel workers, and having each parallel worker saves the model states it owns incurs no additional synchronization overhead and does not negatively impact the overall training speed.

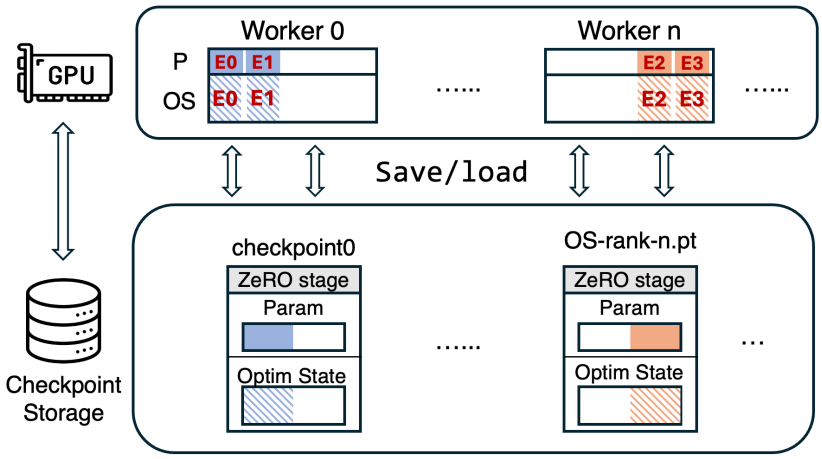

Figure 3: Existing distributed checkpoints are highly coupled to expert parallelism strategy. Each worker saves its own unique sharded MoE parameters (P) and op- timizer states (OS). Reconfiguring Expert Paralleism requires altering each sharded parameter and optimizer states

However, this coupled design introduces significant challenges in reconfiguring checkpoints for different parallelism strategies, necessitating manual development of conversion scripts between parallelization schemes. Consequently, existing frameworks offer limited support for parallelism reconfiguration, typically restricted to basic operations such as modifying the data parallel degree in distributed data parallel implementations Contributors [2023] or performing weight-only conversions for inference scenarios NVIDIA [2024]. While Universal Checkpointing has demonstrated potential as a unified framework supporting arbitrary parallelism strategies, its current implementation focuses on dense models and lacks support for modifying expert parallelism in MoE architectures.

As shown in Figure 3, the challenge of modifying expert parallelism degrees in MoE models stems from the complex partitioning of model states across both GPUs and experts. Consider a 2-expert MoE model where each GPU maintains the model states for one expert while storing optimizer states for all experts. Transitioning from 2 to 4 experts requires splitting and redistributing optimizer states across GPUs, introducing additional complexity in state management.

The need for flexible expert parallelism degree adjustment arises in several critical scenarios. In multi-node training environments, where the number of GPUs per node remains constant but the total node count may vary dynamically, expert parallelism must adapt to maintain optimal resource utilization. Similarly, inference scenarios may require adjusting expert parallelism to match available GPU resources—for instance, consolidating a 2-expert model onto a single GPU for deployment.

4 Design and Implementation

Our design builds upon the Universal Checkpointing framework in DeepSpeed, extending its capabilities to support expert parallelism reconfiguration in Mixture-of-Experts (MoE) models.

4.1 Extend the Pattern Set

Universal Checkpointing enables flexible parallelism reconfiguration through pattern-based transformations of distributed checkpoint files. The effectiveness of this approach hinges on the careful

design and selection of patterns. Optimal patterns must exhibit two essential properties: coverage and flexibility. Coverage ensures comprehensive mapping between distributed checkpoints across diverse parallelism strategies and model architectures, while flexibility enables seamless parallelism reconfiguration operations.

To facilitate expert parallelism reconfiguration, we designed patterns that comprehensively capture sharded model states and optimizer states in MoE models. These patterns must identify both the sharded states for individual experts and the inter-GPU expert mappings.

In our MoE model architecture, each expert possesses two identifiers: a global expert ID and a local expert ID. The global expert ID serves as a unique identifier across all GPUs, while the local expert ID identifies experts within a specific GPU. The relationship between these identifiers is represented as a mapping matrix, where rows correspond to GPUs and columns to experts.

Since checkpoints only store local expert IDs, we developed a pattern that can identify local expert IDs and map them to their global counterparts, while also tracking sharded model and optimizer states for each expert.

To address these requirements, we designed the Unique-MoE pattern.

4.2 Pattern-Aware Reconfiguration Operators

Based on Uniquie-MoE pattern, we developed two key operators:

1. L2G-ID (Local-to-Global ID): This operator maps local expert IDs to global expert IDs through a lookup in the mapping matrix. It serves to decouple checkpoints from specific expert parallelism strategies.

2. G2L-ID (Global-to-Local ID): This operator performs the reverse mapping, converting global expert IDs to local expert IDs during the loading process.

5 Evaluation

We implemented UCP-EP in DeepSpeed and evaluate UCP-EP through a series of experiments on training LLMs.

5.1 Evaluation Methodology

Workloads. For the accuracy evaluation, we focus on evaluating Mixtral-style Transformer based models. We use a subset of the Pile dataset Gao et al. [2020] for training to evaluate the impact to the training loss with and without reconfigured parallelism from UCP-EP.

Hardware. We conducted our experiments on: 4xA100 80GB GPUs (256GB DRAM, 10TB storage, 200Gbps interconnect).

5.2 Decrease the Expert Parallelism Degree

To evaluate UCP-EP's capability of resuming training with reduced GPU count and expert parallelism degree, we first trained the MoE model with EP=4. Given resource constraints, we conducted the experiment for 200 iterations. At the 100th iteration, we enabled UCP on the saved checkpoints and resumed training with different parallelism configurations and GPU counts. We tracked the language model loss throughout the process. As demonstrated in Figure 4, the training successfully resumed with EP=1 parallelism strategies without any degradation in performance.

5.3 Increase the Expert Parallelism Degree

We evaluated UCP-EP's ability to scale up training by increasing both GPU count and expert parallelism degree. Starting with an MoE model trained using EP=2, we ran the experiment for 200 iterations under resource constraints. At iteration 100, we applied UCP to the saved checkpoints and resumed training with expanded parallelism configurations and additional GPUs. Throughout this process, we monitored the language model loss. As shown in Figure 5, the training successfully scaled up to EP=4 while maintaining model performance.

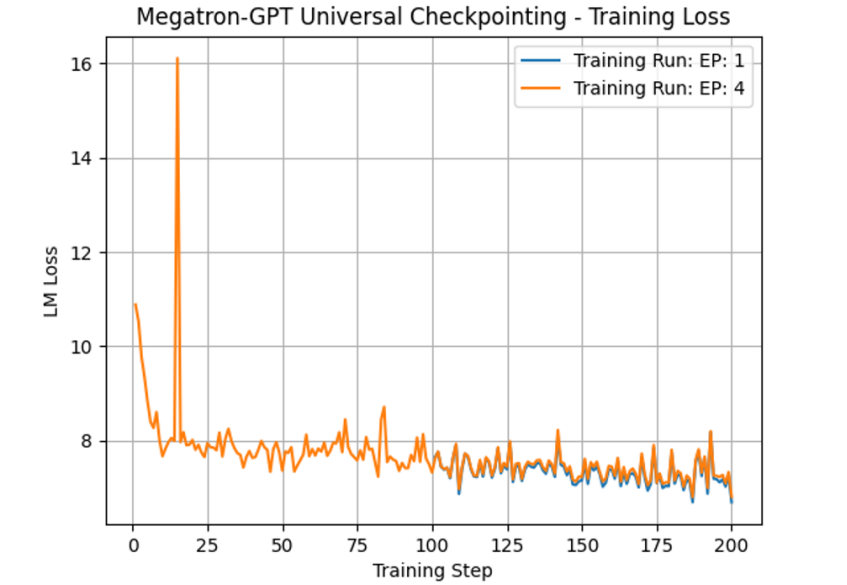

Figure 4: Training curves of decrease EP degree.

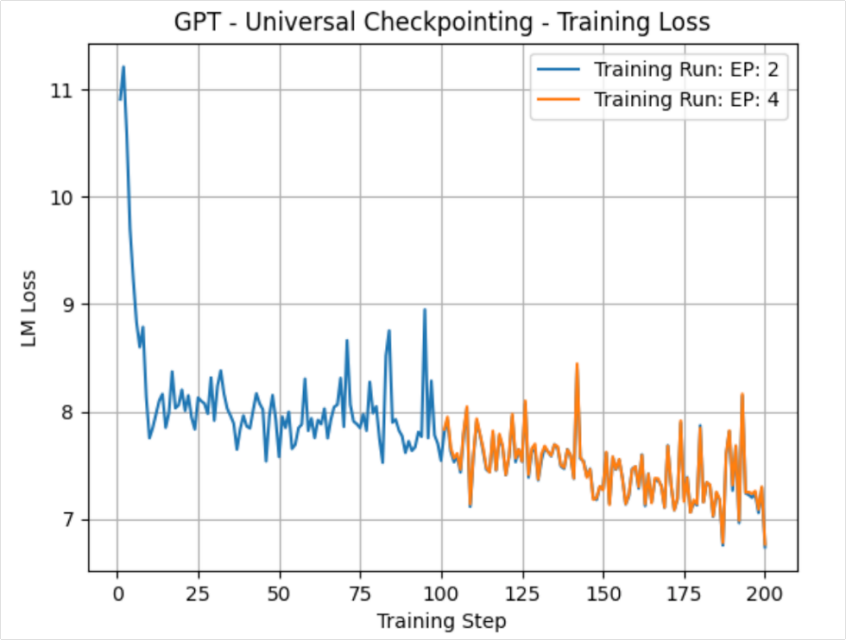

Figure 5: Training curves of increase EP degree.

5.4 Change to other parallelsim strategy

To demonstrate UCP-EP's flexibility in switching between different parallelism strategies, we conducted experiments transitioning from expert parallelism to data parallelism. We initialized training with an MoE model using EP=2 and ran for 200 iterations under our resource constraints. At the midpoint (iteration 100), we applied UCP to the checkpoints and reconfigured the training setup to use data parallelism. The experiment successfully transitioned to DP=4 while preserving model performance, as evidenced by the consistent loss values shown in Figure 6.

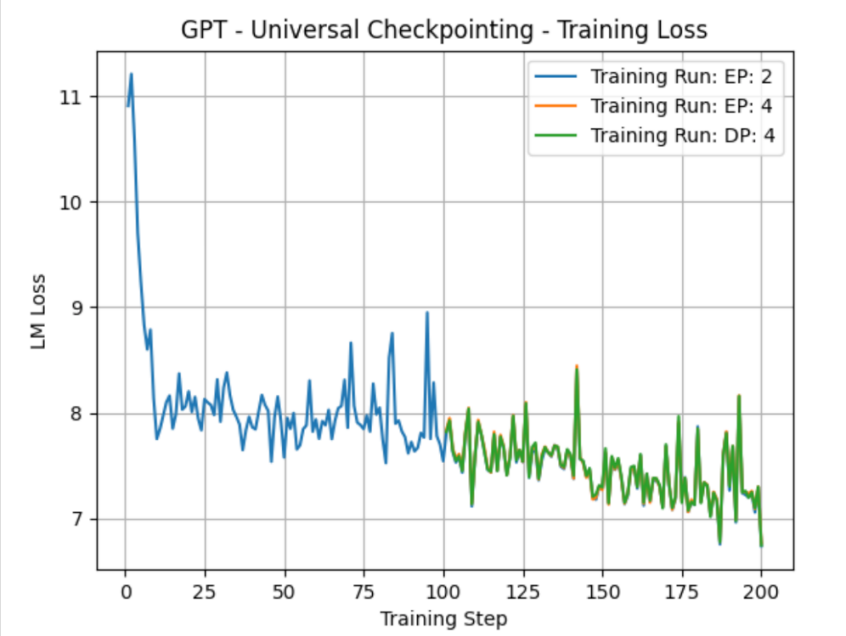

Figure 6: Training curves of changing from EP to DP.

6 Conclusion

We have introduced UCP-EP, a pattern-based extension to DeepSpeed's Universal Checkpointing that addresses the longstanding challenge of reconfiguring expert parallelism in MoE models. By defining a Unique-MoE pattern to capture both local and global expert identifiers and implementing two lightweight mapping operators, UCP-EP fully automates the transformation of distributed checkpoints across arbitrary expert layouts. Our empirical evaluation on Mixtral-style Transformers shows that training can be paused and resumed with fewer or more experts—or even switched to pure data parallelism—without any degradation in model loss.

By decoupling checkpoint formats from expert parallelism strategies, UCP-EP unlocks more agile experimentation and resource utilization for MoE-based LLMs. We believe that this capability will accelerate research and simplify production deployments of next-generation trillion-parameter models.

References

Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. Shortcut-connected expert parallelism for accelerating mixture-of-experts, 2024. URL https://arxiv.org/abs/2404.05019.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, page 1–20, 2025. ISSN 2326-3865. doi: 10.1109/tkde.2025.3554028. URL http://dx.doi.org/10.1109/TKDE.2025.3554028.

PyTorch Contributors. PyTorch Distributed Checkpoint - torch.distributed.checkpoint. https://pytorch.org/docs/stable/distributed.checkpoint.html, 2023.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Seokjin Go and Divya Mahajan. Moetuner: Optimized mixture of expert serving with balanced expert placement and token routing, 2025. URL https://arxiv.org/abs/2502.06643.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024a. URL https://arxiv.org/abs/2401.04088.

Chenyu Jiang, Ye Tian, Zhen Jia, Shuai Zheng, Chuan Wu, and Yida Wang. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping, 2024b. URL https://arxiv.org/abs/2404.19429.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, and et al. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *arXiv preprint arXiv:2006.15704*, 2020.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, and et al. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. *arXiv preprint arXiv:2104.04473*, 2021.

NVIDIA. Megatron-LM: Evaluation and Tasks. https://github.com/NVIDIA/Megatron-LM?tab=readme-ov-file#evaluation-and-tasks, 2024.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory Optimization Towards Training A Trillion Parameter Models. *arXiv preprint arXiv:1910.02054*, 2019.

Jinghan Yao, Quentin Anthony, Aamir Shafi, Hari Subramoni, Dhabaleswar K., and Panda. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference, 2024. URL https://arxiv.org/abs/2401.08383.