Tailoring LLMs for Online Shopping: A Strategy for Multi-Task Learning and Domain-Specific Performance Enhancement

Liu Yankai*†
liuyankai@chinamobile.com
China Mobile Research Institute
Beijing, China

Hao Yifan*
haoyifan@chinamobile.com
China Mobile Research Institute
Beijing, China

Guo Ruipu*
guoruipu@chinamobile.com
China Mobile Research Institute
Beijing, China

Cui Zhaojun*
cuizhaojun@chinamobile.com
China Mobile Research Institute
Beijing, China

Abstract

To address the variety of tasks involved in online shopping, ranging from browsing to making a purchase, there is a necessity for multi-task learning models capable of leveraging shared knowledge across different tasks. Large Language Models (LLMs) have the potential to revolutionize the approach to handling multiple tasks by processing them within a single model and adapting to different prompts. Consequently, Amazon has launched the KDD Cup 2024 Multi-Task Online Shopping Challenge for LLMs competition. In this paper, we present a comprehensive solution that encompasses data processing, model training, in-context learning, acceleration of model inference, and post-processing.Due to the requirements of the competition, we chose the open-source Qwen2-72B as the base model. Our solution has demonstrated remarkable effectiveness in the realm of online shopping. Finally, our team secured 3rd place in the KDD CUP 2024 Task 5 and 4th place in KDD CUP 2024 Task 1 and Task 3.

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

Recommendation, Large Language Model

ACM Reference Format:
Liu Yankai, Hao Yifan, Guo Ruipu, and Cui Zhaojun. 2018. Tailoring LLMs for Online Shopping: A Strategy for Multi-Task Learning and Domain-Specific Performance Enhancement. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

*Equivalent contributions.
†Corresponding Author.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

1.1 Background

Online shopping is a complex process that covers a series of behaviors from navigating through a web of products, reviews, and prices to clicking and purchasing products. This not only requires the model to fully understand product information, but also to insight into customer intentions and preferences. And try to make the best decision based on the explicit or implicit feedback behaviors of the user. However, many current models are task-specific, making it difficult to meet all of these requirements simultaneously.

Large Language Models (LLMs) possess the capabilities of multi-task and few-shot learning, which allows them to handle multiple tasks through a single model with minor prompt adjustments. Therefore, LLMs have the potential to master such complexities of online shopping, and they can also enhance customer experience by providing interactive and timely recommendations.

We proposed a solution that utilizes the capabilities of LLMs to optimize the user's online shopping experience. Utilizing this solution, we participated in the KDD Cup 2024 on Multi-Task Online Shopping Challenge for LLMs hosted by Amazon [6], and achieved commendable results on ShopBench.

1.2 Datasets And Tasks Description

The ShopBench dataset [6] is a multi-task dataset sampled from real-world Amazon shopping data with 57 tasks and approximately 20,000 questions, which can be used as a comprehensive benchmark to simulate the complexities of real-world online shopping. To better mimic real-world scenarios, where you never know the customer's questions beforehand, the organizer only made a small portion of the ShopBench dataset public as the development set, which we use as an offline validation set.

In addition, we also used the publicly available ECInstruct dataset [9], which is an instruction tuning dataset based on Amazon's raw data. It includes 10 e-commerce tasks, which fall within four categories: product understanding, user understanding, query product matching and product Q&A. Each task involves 6 high-quality and diverse instructions, one of which is a clear and concise seed instruction written by humans, and the other five are instructions generated by GPT-4 that are synonymous with the seed instruction but have a different writing style (e.g., wording). Each data sample comprises an instruction, an input, and an output. We calculate that the

ECInstruct dataset has a total of about 264,000 samples(including train and test).

Four different tasks were introduced with the ShopBench dataset:

(1) Shopping Concept Understanding: Emphasizes the ability of LLMs to understand and answer questions related to specific concepts in the field of online shopping, such as brands, product lines, etc.
(2) Shopping Knowledge Reasoning: Focuses on evaluating the model's reasoning ability on products or product attributes with domain-specific implicit knowledge.
(3) User Behavior Alignment: The main goal is to align with heterogeneous and implicit shopping behaviors including browsing, purchasing, query-then-clicking, etc.
(4) Multilingual Abilities: Emphasize the performance of a single model in different shopping locales without re-training, including multi-lingual concept understanding and user behavior alignment.

Besides the above four tasks, the organizer has also set up an all-around task 5, which includes the previous four tasks, aiming to encourage more versatile and comprehensive solutions. We mainly focus on task 1, task 3, and task 5.

2 Methodology

The proposed methodology is segmented into several stages. Initially, we select the Qwen2-72B model as foundational model. Subsequently, we employ the ecinstruct dataset to conduct efficient parameter fine-tuning based on the LoRA technique. Following a comprehensive suite of offline evaluations and model validations, we select the model that performs optimally under the offline evaluation metrics. In the context of the inference submission process, constrained by the computational resources of the competition environment, we proceed to quantize the trained model to ensure operation within the resource constraints of 4*16G T4 GPUs. After quantization, we further enhance the model's efficacy by employing the few-shot approach during inference. An increase in input length leads to an extended response time for the initial character, hence, to ensure completion within the allotted time, we utilize the prefix caching method to augment the inference speed, thereby guaranteeing that reasoning is completed within the required timeframe. Upon the completion of model inference, we apply post-processing techniques to refine the model's output results, thereby enhancing the accuracy of the output format.

2.1 Base Model

With the emergence of ChatGPT[8], the popularity of LLMs has gradually increased worldwide. A growing number of LLMs are pursuing advancements similar to those made by the OpenAI GPT family. With a free and open attitude, many models with superior performance, including Llama[11], Qwen[1], etc., have been released in the open-source community.

The Qwen2[15] series models, as excellent models in the open source community, are at the forefront of the major evaluation lists. Compared to previous qwen series models, qwen2 has been continuously pre-trained on a high-quality corpus of more than 7 trillion tokens. Both the quality and quantity of the code and math sections have been improved, which not only enhances the code and math capabilities, but also its reasoning capabilities. In the alignment stage, all models underwent supervised fine-tuning and direct preference optimization, and have good ability to follow human instructions. Specifically, Qwen2-72B-Instruct[15] showcases remarkable performance: 9.1 on MT-Bench, 48.1 on Arena-Hard, and 35.7 on LiveCodeBench. The comparison with other models is shown in table1. In addition, Qwen2 demonstrates powerful multilingual capabilities and performs substantially better than GPT-3.5-Turbo.

Table 1: Performance of 70B+ instruction-tuned models

Datasets	Llama-3-70B	Qwen1.5-110B	**Qwen2-72B**
MT-Bench	8.95	8.88	**9.12**
Arena-Hard	41.1	39.8	**48.1**
LiveCodeBench	29.3	25.3	**35.7**

Nevertheless, it is noted that inference under BF16 format for Qwen2-72B-Instruct demands approximately 144GB of gpu memory. This amount exceeds the competition limit of 16GB * 4 = 64G. Therefore, we contemplate employing model compression techniques to lower the resource consumption during inference. We adopted the GPTQ[4] quantification method, which decreased the gpu memory to 48G. Thus, we were able to select Qwen2-72B-Instruct as the base model.

2.2 Datasets

Our solution leverages the ECInstruct dataset [9] for supervised fine-tuning. ECInstruct is a comprehensive e-commerce instruction dataset containing more than 260,000 samples. It encompasses 10 diverse tasks, each with 6 different instructions, ensuring rich instruction and task variety.

To tailor the ECInstruct dataset to our specific needs, we applied a series of preprocessing steps including filtering, downsampling, and reformatting. Firstly, we categorized the tasks in ECInstruct based on the taxonomy defined in this challenge, as depicted in Table 2. While the majority of tasks share a similar format with our challenge, we noticed that binary classification tasks, which require the model to make a binary judgment and output either yes or no, deviate a lot from the format of the tasks in our challenge. Consequently, we excluded these three binary classification tasks from ECInstruct. Subsequently, we randomly selected 200,000 samples from the filtered dataset for fine-tuning purposes.

Moreover, in order to fully leverage the power of SFT, we made adjustments to the format of samples in ECInstruct to align with those in this challenge. For instance, we've replaced the letter options in multiple-choice questions (A, B, C, ...) with numerical options (0, 1, 2, ...). Additionally, if a task in ECInstruct shares a similar semantic with a task in the development set, we would swap the instructions with those from the development task. For example, we replaced the instructions for Sentiment Analysis with those for task 15 in development set, which is also a sentiment analysis task. Multiclass Product Classification and task 11 both use ECSI labels, so we also replaced instructions for Multiclass Product Classification with those for task 11. We then combine the reformatted data with the original data to construct our final sft dataset.

Table 2: Task Types in ECInstruct

Task Type	Task Name
named entity generation	Attribute Value Extraction
multiple choice	Multiclass Product Classification
	Product Relation Prediction
	Sentiment Analysis
retrieval	Sequential Recommendation
ranking	Query Product Rank
generation	Answer Generation
binary classification	Answerability Prediction
	Product Substitute Identification
	Product Matching

2.3 Supervised Fine-Tuning

Open source LLMs possess excellent generic capabilities and instruction compliance capabilities. However, they frequently fail to achieve the state-of-the-art (SOTA) effect on specific downstream tasks. Hence, it is necessary to fine-tune the general LLMs based on the specific task. As a result, choosing the appropriate framework and fine-tuning method is of crucial importance.

Fine-Tuning Framework For efficient fine-tuning of LLMs, we chose LLAMAFACTORY[17], a framework that promotes the democratization of the fine-tuning of LLMs. It integrates various efficient fine-tuning methods through scalable modules, allowing the fine-tuning of hundreds of LLMs with the least resources and high throughput. Additionally, it simplifies commonly used training approaches, such as generative pre-training, supervised fine-tuning (SFT), reinforcement learning from human feedback (RLHF), and direct preference optimization (DPO). We are able to utilize command-line or web interfaces to customize and fine-tune LLMs with little or no coding at all.

- **Efficient Training**: the framework supports state-of-the-art efficient fine-tuning methods, including LoRA[5], QLoRA[3] and GaLore[16]. It also applys the trainers of Transformers for pre-training and SFT, while utilizing the trainers of TRL[12] for RLHF and DPO.
- **Distributed Training**: the framework supports DeepSpeed[10] for distributed training, the memory consumption can be further reduced.

Parameter-Efficient Fine-Tuning In order to fine-tune the model more effectively for downstream tasks without losing most of the power of the LLMs, we chose a efficient fine-tuning method: LoRA[5]. Instead of directly training some dense layers in a neural network, it can indirectly train them by optimizing the rank decomposition matrices of the changes in these dense layers during adaptation, while keeping the pre-trained weights unchanged.

Model Quantization In order to compress our base model, we choose the GPTQ-Int4[4] quantization method, which can significantly reduce the gpu memory requirements for model inference. GPTQ is capable of quantizing GPT models having 175 billion parameters within approximately four GPU hours. It can reduce the bitwidth to 3 or 4 bits per weight, while the accuracy degradation is negligible compared to the uncompressed baseline.

2.4 In-Context Learning

To further enhance the performance of our model during the inference stage, we employ the paradigm of In-Context Learning (ICL) [2]. By providing a few relevant demonstrations as context, ICL can stimulate LLM's domain-specific abilities without further parameter updates.

ICL Strategy and Implementation Our ICL strategy involves meticulous selection and utilization of few-shot examples to guide the model's behavior. We utilize the development set for offline validation, employing the results to guide our selection for few-shot examples.

The key aspect of our approach is the dynamic selection of few-shot examples based on the task type. We discern between multiple-choice tasks and other task types to tailor our ICL demonstrations accordingly.

- **Multiple-Choice Tasks**: For multiple-choice tasks, we identify one relevant example from the development set and use it as a one-shot demonstration. This focused approach ensures that the model learns the special pattern and semantics of multiple-choice tasks effectively.
- **Other Task Types**: For tasks beyond the multiple-choice category, we adopt a more comprehensive approach. We select one representative example from each task, ensuring a diverse range of demonstrations. These examples are then combined to form our few-shot examples, providing the model with a broader understanding of various tasks.

This task-specific selection of few-shot examples empowers our model to adapt and generalize effectively across different task types, resulting in improved performance during the inference stage.

Finally, our final prompt is constrcucted using the chat template of Qwen2, where the few-shot examples are added as history dialogue.

Inference Acceleration Given the time constraints of the challenge, we leveraged vLLM [7] to accelerate inference. vLLM is an open-source library designed for rapid LLM inference and serving, offering significant performance improvements. To further accelerate the generation, we employed fixed demonstrations for each task type, eliminating the need for dynamically choosing demonstrations during inference. Additionally, this approach allows us to utilized the prefix-caching technique to expedite the computation for the demonstration part. Prefix-caching would store the KV cache for previously executed queries. Consequently, when a fresh query has a matching prefix with any of the stored queries, it can immediately utilize the existing KV cache and avoid recalculating the common part. As a result, the overhead introduced by ICL is minimal and can be largely ignored.

2.5 Post-process

During the evaluation of our solution on the development set, we encountered several instances of failures stemming from the generation of responses with invalid formats. To rectify this issue, we implemented a post-processing step to refine our outputs.

One prevalent issue we observed was the generation of duplicate indices in ranking tasks. To mitigate this, we analyzed the output patterns across the five task types and identified the distinctive patterns associated with ranking tasks. Subsequently, we separated

Table 3: Overall Performance

Track	Score	Rank
track1	0.823	4th
track3	0.722	4th
track5	0.773	3rd

the ranking tasks from other types of tasks based on these patterns. We then proceeded to deduplicate the generated lists for ranking tasks and appended any missing indices to the end of the sequence. Another challenge we encountered was the tendency of LLM to engage in reasoning prior to generating the final choice. To circumvent this, we leveraged the LLM's instruction following capabilities and introduced additional instructions to guide it towards directly generating the desired output without engaging in reasoning.

In addition to our primary solution, we conducted experiments to explore the potential of employing the Chain of Thought (CoT) strategy [13] to enhance the accuracy of our solution for multiple-choice tasks. Our CoT strategy involves instructing the LLM to provide a brief analysis or reasoning before generating the final answer. However, this approach also increased the risk of parsing failures. To address this challenge, we utilized outlines [14], an open-source tool that offers methods for controlling the generation of language models, making their outputs more predictable. We employed outlines to guide the generation of responses for multiple-choice tasks to conform to the following regex pattern:

```
r"[A-Za-z][^\|]{10,100}\|\d"
```

This pattern specifies that the response should begin with a brief analysis or reasoning (with a length between 10 and 100 characters and starts with a letter), and conclude with a delimiter ("|") and the corresponding option number. By applying this pattern, we were able to reliably extract the final answer by splitting the output with the delimiter, while maintaining the benefits of the CoT strategy.

3 Experiments

The performance of our models is listed in Table3. The model of track1, having been trained on the ECInstruct-20w dataset with a lora rank of 64 and applying track1-fewshot for inference, got a score of 0.823 and was ranked 4th. The model of track3, having not been trained and applying mix-fewshot for inference, got a score of 0.722 and was ranked 4th. The model of track5, having been trained on the ECInstruct-20w dataset with a lora rank of 64 and applying fewshot for inference, got a score of 0.773 and was ranked 3rd.

4 Conclusion

In this paper, we elaborate on the comprehensive solutions across the entire pipeline, including data processing, model training, in-context learning, acceleration of model inference, and post-processing. In our approach to model data processing, we enhance model efficacy by refining the ecinstruct dataset and employing techniques such as efficient parameter fine-tuning and few-shot learning. The evaluation results validate the effectiveness of the methods, securing 3rd place in KDD CUP 2024 Task 5 and 4th in KDD CUP 2024 Task 1 and Task 3. Moving forward, this methodology has the potential to be applied to large-scale online shopping platforms, further enhancing the user experience on such websites, much like a knowledgeable shopping assistant in real life.

References

[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. arXiv:2309.16609 [cs.CL] https://arxiv.org/abs/2309.16609

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 10088–10115. https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf

[4] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323 [cs.LG] https://arxiv.org/abs/2210.17323

[5] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nZeVKeeFYf9

[6] Amazon Rufus Team kddcup2024. 2024. Amazon Kdd Cup 2024 Multi Task Online Shopping Challenge for LLMs. https://www.aicrowd.com/challenges/amazon-kdd-cup-2024-multi-task-online-shopping-challenge-for-llms

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[8] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] https://arxiv.org/abs/2203.02155

[9] Bo Peng, Xinyi Ling, Ziru Chen, Huan Sun, and Xia Ning. 2024. eCeLLM: Generalizing Large Language Models for E-commerce from Large-scale, High-quality Instruction Data. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=LWRI4uPG2X

[10] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL] https://arxiv.org/abs/2307.09288

[12] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. TRL: Transformer Reinforcement Learning. https://github.com/huggingface/trl.

[13] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1800, 14 pages.

[14] Brandon T Willard and Rémi Louf. 2023. Efficient Guided Generation for LLMs. *arXiv preprint arXiv:2307.09702* (2023).

[15] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. arXiv:2407.10671 [cs.CL] https://arxiv.org/abs/2407.10671

[16] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. arXiv:2403.03507 [cs.LG] https://arxiv.org/abs/2403.03507

[17] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. arXiv:2403.13372 [cs.CL] https://arxiv.org/abs/2403.13372