LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models

Yaowei Zheng, Richong Zhang*, **Junhao Zhang, Yanhan Ye, Zheyan Luo**
School of Computer Science and Engineering, Beihang University
Open-source repository: **https://github.com/hiyouga/LLaMA-Factory**
Demonstration video: **https://youtu.be/W29FgeZEpus**

Abstract

Efficient fine-tuning is vital for adapting large language models (LLMs) to downstream tasks. However, it requires non-trivial efforts to implement these methods on different models. We present LlamaFactory, a unified framework that integrates a suite of cutting-edge efficient training methods. It provides a solution for flexibly customizing the fine-tuning of 100+ LLMs without the need for coding through the built-in web UI LlamaBoard. We empirically validate the efficiency and effectiveness of our framework on language modeling and text generation tasks. It has been released at https://github.com/hiyouga/LLaMA-Factory and received over 25,000 stars and 3,000 forks.

1 Introduction

Large language models (LLMs) (Zhao et al., 2023) present remarkable reasoning capabilities and empower a wide range of applications, such as question answering (Jiang et al., 2023b), machine translation (Wang et al., 2023b; Jiao et al., 2023a), and information extraction (Jiao et al., 2023b). Subsequently, a substantial number of LLMs are developed and accessible through open-source communities. For example, Hugging Face's open LLM leaderboard (Beeching et al., 2023) boasts over 5,000 models, offering convenience for individuals seeking to leverage the power of LLMs.

Fine-tuning extremely large number of parameters with limited resources becomes the main challenge of adapting LLM to downstream tasks. A popular solution is efficient fine-tuning (Houlsby et al., 2019; Hu et al., 2022; Dettmers et al., 2023), which reduces the training cost of LLMs when adapting to various tasks. However, the community contributes various methods for efficient fine-tuning, lacking a systematic framework that adapts and unifies these methods to different LLMs and provides a friendly interface for user customization.

To address the above problems, we develop LlamaFactory, a framework that democratizes the fine-tuning of LLMs. It unifies a variety of efficient fine-tuning methods through scalable modules, enabling the fine-tuning of hundreds of LLMs with minimal resources and high throughput. In addition, it streamlines commonly used training approaches, including generative pre-training (Radford et al., 2018), supervised fine-tuning (SFT) (Wei et al., 2022), reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), and direct preference optimization (DPO) (Rafailov et al., 2023). Users can leverage command-line or web interfaces to customize and fine-tune their LLMs with minimal or no coding effort.

LlamaFactory consists of three main modules: ***Model Loader***, ***Data Worker*** and ***Trainer***. We minimize the dependencies of these modules on specific models and datasets, allowing the framework to flexibly scale to hundreds of models and datasets. Concretely, we first establish a model registry where the *Model Loader* can precisely attach adapters to the pre-trained models by identifying exact layers. Then we develop a data description specification that allows the *Data Worker* to gather datasets by aligning corresponding columns. Furthermore, we provide plug-and-play implementations of state-of-the-art efficient fine-tuning methods that enable the *Trainer* to activate by replacing default ones. Our design allows these modules to be reused across different training approaches, significantly reducing the integration costs.

LlamaFactory is implemented with PyTorch (Paszke et al., 2019) and significantly benefits from open-source libraries, such as Transformers (Wolf et al., 2020), PEFT (Mangrulkar et al., 2022), and TRL (von Werra et al., 2020). On the basis, we provide an out-of-the-box framework with a higher level of abstraction. Additionally, we build LlamaBoard with Gradio (Abid et al., 2019), enabling fine-tuning LLMs with no coding efforts required.

*Corresponding author

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 400–410
August 11-16, 2024

	LLAMAFACTORY	FastChat	LitGPT	LMFlow	Open-Instruct
LoRA	✓	✓	✓	✓	✓
QLoRA	✓	✓	✓	✓	✓
DoRA	✓				
LoRA+	✓				
PiSSA	✓				
GaLore	✓	✓		✓	✓
BAdam	✓				
Flash attention	✓	✓	✓	✓	✓
S^2 attention	✓				
Unsloth	✓		✓		
DeepSpeed	✓	✓	✓	✓	✓
SFT	✓	✓	✓	✓	✓
RLHF	✓			✓	
DPO	✓				✓
KTO	✓				
ORPO	✓				

Table 1: Comparison of features in LLAMAFACTORY with popular frameworks of fine-tuning LLMs.

LLAMAFACTORY is open-sourced under the Apache-2.0 license. It has already garnered over 25,000 stars and 3,000 forks on the GitHub, and hundreds of open-source models have been built upon LLAMAFACTORY on the Hugging Face Hub[1]. For example, Truong et al. (2024) build GemSUra-7B based on LLAMAFACTORY, revealing the cross-lingual abilities of Gemma (Mesnard et al., 2024). Furthermore, dozens of studies have utilized our framework to explore LLMs (Wang et al., 2023a; Yu et al., 2023; Bhardwaj et al., 2024).

2 Related Work

With the rapid increase in demand for fine-tuning LLMs, numerous frameworks for adapting LLMs to specific purposes have been developed. LLaMA-Adapter (Zhang et al., 2024) efficiently fine-tunes the Llama model (Touvron et al., 2023a) using a zero-initialized attention. FastChat (Zheng et al., 2023) is a framework focused on training and evaluating LLMs for chat completion purposes. LitGPT (AI, 2023) provides the implementation of generative models and supports various training methods. Open-Instruct (Wang et al., 2023c) provides recipes for training instruct models. Colossal AI (Li et al., 2023b) takes advanced parallelism strategies for distributed training. LMFlow (Diao et al., 2024) supports training LLMs for specialized domains or tasks. GPT4All (Anand et al., 2023) allows LLMs to run on consumer devices, while also providing fine-tuning capabilities. Compared with existing competitive frameworks, LLAMAFACTORY supports a broader range of efficient fine-tuning techniques and training approaches. We list the features among representative frameworks in Table 1.

[1] https://huggingface.co/models?other=llama-factory

	Freeze-tuning	GaLore	LoRA	DoRA	LoRA+	PiSSA
Mixed precision	✓	✓	✓	✓	✓	✓
Checkpointing	✓	✓	✓	✓	✓	✓
Flash attention	✓	✓	✓	✓	✓	✓
S^2 attention	✓	✓	✓	✓	✓	✓
Quantization	✗	✗	✓	✓	✓	✓
Unsloth	✗	✗	✓	✓	✓	✓

Table 2: Compatibility between the fine-tuning techniques featured in LLAMAFACTORY.

3 Efficient Fine-Tuning Techniques

Efficient LLM fine-tuning techniques can be divided into two main categories: those focused on optimization and those aimed at computation. The primary objective of efficient optimization techniques is to fine-tune the parameters of LLMs while keeping costs to a minimum. On the other hand, efficient computation methods seek to decrease the time or space for the required computation in LLMs. The methods included in LLAMAFACTORY are listed in Table 2. We will present these efficient fine-tuning techniques and show the substantial efficiency improvement achieved by incorporating them into our framework in the following sections.

3.1 Efficient Optimization

Firstly, we provide an overview of the efficient optimization techniques utilized in LLAMAFACTORY. The freeze-tuning method (Houlsby et al., 2019) involves freezing a majority of parameters while fine-tuning the remaining parameters in a small subset of decoder layers. Another method called gradient low-rank projection (GaLore) (Zhao et al., 2024) projects gradients into a lower-dimensional space, facilitating full-parameter learning in a memory-efficient manner. Similarly, BAdam (Luo et al., 2024) leverages block coordinate descent (BCD) to efficiently optimize the extensive parameters. On the contrary, the low-rank adaptation (LoRA) (Hu et al., 2022) method freezes all pre-trained weights and introduces a pair of trainable low-rank matrices to the designated layer. When combined with quantization, this approach is referred to as QLoRA (Dettmers et al., 2023), which additionally reduces the memory usage. DoRA (Liu et al., 2024) breaks down pre-trained weights into magnitude and direction components and updates directional components for enhanced performance. LoRA+ (Hayou et al., 2024) is proposed to overcome the sub-optimality of LoRA. PiSSA (Meng et al., 2024) initializes adapters with the principal components of the pre-trained weights for faster convergence.

3.2 Efficient Computation

In LLAMAFACTORY, we integrate a range of techniques for efficient computation. Commonly utilized techniques encompass mixed precision training (Micikevicius et al., 2018) and activation checkpointing (Chen et al., 2016). Drawing insights from the examination of the input-output (IO) expenses of the attention layer, flash attention (Dao et al., 2022) introduces a hardware-friendly approach to enhance attention computation. S^2 attention (Chen et al., 2024a) tackles the challenge of extended context with shifted sparse attention, thereby diminishing memory usage in fine-tuning long-context LLMs. Various quantization strategies (Dettmers et al., 2022a; Frantar et al., 2023; Lin et al., 2023; Egiazarian et al., 2024) decrease memory requirements in large language models (LLMs) by utilizing lower-precision representations for weights. Nevertheless, the fine-tuning of quantized models is restricted to the adapter-based techniques like LoRA (Hu et al., 2022). Unsloth (Han and Han, 2023) incorporates Triton (Tillet et al., 2019) for implementing the backward propagation of LoRA, which reduces floating-point operations (FLOPs) during gradient descent and leads to expedited LoRA training.

LLAMAFACTORY seamlessly combines these techniques into a cohesive structure to enhance the efficiency of LLM fine-tuning. This results in a reduction of the memory footprint from 18 bytes per parameter during mixed precision training (Micikevicius et al., 2018) or 8 bytes per parameter in half precision training (Le Scao et al., 2022) to only 0.6 bytes per parameter. Further elaboration on the components in LLAMAFACTORY will be provided in the subsequent section.

4 LLAMAFACTORY Framework

LLAMAFACTORY consists of three main modules: *Model Loader*, *Data Worker*, and *Trainer*. The *Model Loader* manipulates various model architectures for fine-tuning, supporting both large language models (LLMs) and vision language models (VLMs). The *Data Worker* processes data from different tasks through a well-designed pipeline, supporting both single-turn and multi-turn dialogues. The *Trainer* applies efficient fine-tuning techniques to different training approaches, supporting pre-training, instruction tuning and preference optimization. Beyond that, LLAMABOARD provides a friendly visual interface to access these modules, enabling users to configure and launch individual LLM fine-tuning instance codelessly and monitor the training status synchronously. We illustrate the relationships between these modules and the overall architecture of LLAMAFACTORY in Figure 1.

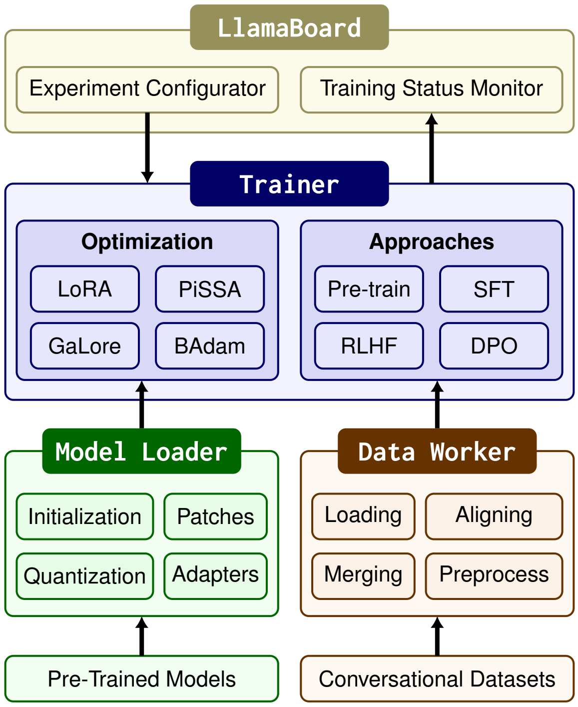

Figure 1: The architecture of LLAMAFACTORY.

4.1 Model Loader

This section initially presents the four components in *Model Loader*: model initialization, model patching, model quantization, and adapter attaching, followed by a description of our approach of adapting to a wide range of devices by handling the parameter floating-point precision during fine-tuning.

Model Initialization We utilize the *Auto Classes* of Transformers (Wolf et al., 2020) to load pre-trained models and initialize parameters. Specifically, we load the vision language models using the `AutoModelForVision2Seq` class while the rest are loaded using the `AutoModelForCausalLM` class. The tokenizer is loaded using the `AutoTokenizer` class along with the model. In cases where the vocabulary size of the tokenizer exceeds the capacity of the embedding layer, we resize the layer and initialize new parameters with noisy mean initialization. To determine the scaling factor for RoPE scaling (Chen et al., 2023), we compute it as the ratio of the maximum input sequence length to the context length of the model.

Model Patching To enable the S^2 attention, we employ a monkey patch to replace the forward computation of models. However, we use the native class to enable flash attention as it has been widely supported since Transformers 4.34.0. To prevent excessive partitioning of the dynamic layers, we set the mixture-of-experts (MoE) blocks as leaf modules when we optimize the MoE models under DeepSpeed ZeRO stage-3 (Rasley et al., 2020).

Model Quantization Dynamically quantizing models to 8 bits or 4 bits with LLM.int8 (Dettmers et al., 2022a) can be performed through the bitsandbytes library (Dettmers, 2021). For 4-bit quantization, we utilize the double quantization and 4-bit normal float as QLoRA (Dettmers et al., 2023). We also support fine-tuning the models quantized by the post-training quantization (PTQ) methods, including GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2023), and AQLM (Egiazarian et al., 2024). Note that we cannot directly fine-tune the quantized weights; thus, the quantized models are only compatible with adapter-based methods.

Adapter Attaching We automatically identify the appropriate layers to attach adapters through traversing the model layers. The low-rank adapters are attached to all the linear layers for a better convergence as suggested by (Dettmers et al., 2023). The PEFT (Mangrulkar et al., 2022) library provides an extremely convenient way to implement the adapter-based methods such as LoRA (Hu et al., 2022), rsLoRA (Kalajdzievski, 2023), DoRA (Liu et al., 2024) and PiSSA (Meng et al., 2024). We replace the backward computation with the one of Unsloth (Han and Han, 2023) to accelerate the training. To perform reinforcement learning from human feedback (RLHF), a value head layer is appended on the top of the transformer model, mapping the representation of each token to a scalar.

Precision Adaptation We handle the floating-point precision of pre-trained models based on the capabilities of computing devices. For NVIDIA GPUs, we adopt bfloat16 precision if the computation capability is 8.0 or higher. Otherwise, float16 is adopted. Besides, we adopt float16 for Ascend NPUs and AMD GPUs and float32 for non-CUDA devices. In mixed precision training, we set all trainable parameters to float32 for training stability. Nevertheless, we retain the trainable parameters as bfloat16 in half precision training.

Plain text	[{"text": "..."}, {"text": "..."}]
Alpaca-like data	[{"instruction": "...", "input": "...", "output": "..."}]
ShareGPT-like data	[{"conversations": [{"from": "human", "value": "..."}, {"from": "gpt", "value": "..."}]}]
Preference data	[{"instruction": "...", "input": "...", "output": ["...", "..."]}]
Standardized data	{"prompt": [{"role": "...", "content": "..."}], "response": [{"role": "...", "content": "..."}], "system": "...", "tools": "...", "images": ["..."]}

Table 3: Dataset structures in LLAMAFACTORY.

4.2 Data Worker

We develop a data processing pipeline, including dataset loading, dataset aligning, dataset merging and dataset pre-processing. It standardizes datasets of different tasks into a unified format, enabling us to fine-tune models on datasets in various formats.

Dataset Loading We utilize the Datasets (Lhoest et al., 2021) library to load the data, which allows the users to load remote datasets from the Hugging Face Hub or read local datasets via scripts or through files. The Datasets library significantly reduces memory overhead during data processing and accelerates sample querying using Arrow (Apache, 2016). By default, the whole dataset is downloaded to local disk. However, if a dataset is too large to be stored, our framework provides dataset streaming to iterate over it without downloading.

Dataset Aligning To unify the dataset format, we design a data description specification to characterize the structure of datasets. For example, the alpaca dataset has three columns: instruction, input and output (Taori et al., 2023). We convert the dataset into a standard structure that is compatible with various tasks according to the data description specification. Some examples of dataset structures are shown in Table 3.

Dataset Merging The unified dataset structure provides an efficient approach for merging multiple datasets. For the datasets in non-streaming mode, we simply concatenate them before the datasets are shuffled during training. However, in streaming mode, simply concatenating the datasets impedes data shuffling. Therefore, we offer methods to alternately read the data from different datasets.

Dataset Pre-processing LLAMAFACTORY is designed for fine-tuning the text generative models, which is primarily used in chat completion. Chat template is a crucial component in these models, because it is highly related to the instruction-following abilities of these models. Therefore, we provide dozens of chat templates that can be automatically chosen according to the model type. We encode the sentence after applying the chat template using the tokenizer. By default, we only compute loss on the completions, while the prompts are disregarded (Taori et al., 2023). Optionally, we can utilize sequence packing (Krell et al., 2021) to reduce the training time, which is automatically enabled when performing generative pre-training.

4.3 Trainer

Efficient Training We integrate state-of-the-art efficient fine-tuning methods, including LoRA+ (Hayou et al., 2024), GaLore (Zhao et al., 2024) and BAdam (Luo et al., 2024) to the *Trainer* by replacing the default components. These fine-tuning methods are independent of the *Trainer*, making them easily applicable to various tasks. We utilize the trainers of Transformers (Wolf et al., 2020) for pre-training and SFT, while adopting the trainers of TRL (von Werra et al., 2020) for RLHF and DPO. We also include trainers of the advanced preference optimization methods such as KTO (Ethayarajh et al., 2024) and ORPO (Hong et al., 2024) from the TRL library. The tailored data collators are leveraged to differentiate trainers of various training approaches. To match the input format of the trainers for preference data, we build $2n$ samples in a batch where the first n samples are chosen examples and the last n samples are rejected examples.

Model-Sharing RLHF Allowing RLHF training on consumer devices is crucial for democratizing LLM fine-tuning. However, it is difficult because RLHF training requires four different models. To address this problem, we propose model-sharing RLHF, enabling entire RLHF training with no more than one pre-trained model. Concretely, we first train an adapter and a value head with the objective function for reward modeling, allowing the model to compute reward scores. Then we initialize another adapter and value head and train them with the PPO algorithm (Ouyang et al., 2022). The adapters and value heads are dynamically switched through the `set_adapter` and `disable_adapter` methods of PEFT (Mangrulkar et al., 2022) during training, allowing a single pre-trained model to serve as policy model, value model, reference model, and reward model simultaneously. To the best of our knowledge, this is the first method that supports RLHF training on consumer devices.

Distributed Training We can combine the above trainers with DeepSpeed (Rasley et al., 2020; Ren et al., 2021) for distributed training. We adopt data parallelism to fully exploit the ability of computing devices. Leveraging the DeepSpeed ZeRO optimizer, the memory consumption can be further reduced via partitioning or offloading.

4.4 Utilities

Model Inference During inference time, we reuse the chat template from the *Data Worker* to build the model inputs. We offer support for sampling the model outputs using Transformers (Wolf et al., 2020) and vLLM (Kwon et al., 2023), both of which support stream decoding. Additionally, we implement an OpenAI-style API that utilizes the asynchronous LLM engine and paged attention of vLLM, to provide high-throughput concurrent inference services, facilitating the deployment of fine-tuned LLMs into various applications.

Model Evaluation We include several metrics for evaluating LLMs, including multiple-choice tasks such as MMLU (Hendrycks et al., 2021), CMMLU (Li et al., 2023a), and C-Eval (Huang et al., 2023), as well as calculating text similarity scores like BLEU-4 (Papineni et al., 2002) and ROUGE (Lin, 2004). This feature facilitates users to measure the abilities of the fine-tuned models.

4.5 LlamaBoard: A Unified Interface for LlamaFactory

LlamaBoard is a unified user interface based on Gradio (Abid et al., 2019) that allows users to customize the fine-tuning of LLMs without writing any code. It offers a streamlined model fine-tuning and inference service, enabling users to easily explore the potential of LLMs in their environments. LlamaBoard has the following notable features.

Easy Configuration LlamaBoard allows us to customize the fine-tuning arguments through interaction with the web interface. We provide default values for a majority of arguments that are recommended for most users, simplifying the configuration process. Moreover, users can preview the datasets on the web UI to validate them.

Monitorable Training During the training process, the training logs and loss curves are visualized and updated in real time, allowing users to monitor the training progress. This feature provides valuable insights to analyze the fine-tuning process.

	Gemma-2B				Llama2-7B				Llama2-13B			
Method	Trainable Params	Memory (GB)	Throughput (Tokens/s)	PPL	Trainable Params	Memory (GB)	Throughput (Tokens/s)	PPL	Trainable Params	Memory (GB)	Throughput (Tokens/s)	PPL
Baseline	/	/	/	11.83	/	/	/	7.53	/	/	/	6.66
Full-tuning	2.51B	17.06	3090.42	10.34	6.74B	38.72	1334.72	5.56	/	/	/	/
Freeze-tuning	0.33B	8.10	5608.49	11.33	0.61B	15.69	2904.98	6.59	0.95B	29.02	1841.46	6.56
GaLore	2.51B	10.16	2483.05	10.38	6.74B	15.43	1583.77	5.88	13.02B	28.91	956.39	**5.72**
LoRA	0.16B	7.91	**3521.05**	**10.19**	0.32B	16.32	**1954.07**	**5.81**	0.50B	30.09	**1468.19**	5.75
QLoRA	0.16B	**5.21**	3158.59	10.46	0.32B	**7.52**	1579.16	5.91	0.50B	**12.61**	973.53	5.81

Table 4: Comparison of the training efficiency using different fine-tuning methods in LLAMAFACTORY. The best result among GaLore, LoRA and QLoRA of each model is in **bold**.

Flexible Evaluation LLAMABOARD supports calculating the text similarity scores on the datasets to automatically evaluate models or performing human evaluation by chatting with them.

Multilingual Support LLAMABOARD provides localization files, facilitating the integration of new languages for rendering the interface. Currently we support three languages: English, Russian and Chinese, which allows a broader range of users to utilize LLAMABOARD for fine-tuning LLMs.

5 Empirical Study

We systematically evaluate LLAMAFACTORY from two perspectives: 1) the training efficiency in terms of memory usage, throughput and perplexity. 2) the effectiveness of adaptation to downstream tasks.

5.1 Training Efficiency

Experimental Setup We utilize the PubMed dataset (Canese and Weis, 2013), which comprises over 36 million records of biomedical literature. We extract around 400K tokens from the abstract of the literature to construct the training corpus. Then we fine-tune the Gemma-2B (Mesnard et al., 2024), Llama2-7B and Llama2-13B (Touvron et al., 2023b) models using the generative pre-training objective with various efficient fine-tuning methods. We compare the results of full-tuning, freeze-tuning, GaLore, LoRA and 4-bit QLoRA. After fine-tuning, we calculate the perplexity on the training corpus to evaluate the efficiency of different methods. We also incorporate the perplexities of the pre-trained models as baselines.

In this experiment, we adopt a learning rate of 10^{-5}, a token batch size of 512. We fine-tune these models using the 8-bit AdamW optimizer (Dettmers et al., 2022b) in bfloat16 precision with activation checkpointing to reduce the memory footprint. In freeze-tuning, we only fine-tune the last 3 decoder layers of the model. For GaLore, we set the rank and scale to 128 and 2.0, respectively. For LoRA and QLoRA, we attach adapters to all linear layers and set the rank and alpha to 128 and 256, respectively. All the experiments are conducted on a single NVIDIA A100 40GB GPU. We enable flash attention in all experiments and Unsloth for LoRA and QLoRA experiments.

Results The results about the training efficiency are presented in Table 4, where memory refers to the peak memory consumed during training, throughput is calculated as the number of tokens trained per second, and PPL represents the perplexity of the model on the training corpus. Since full-tuning Llama2-13B lead to a memory overflow, the results are not recorded. We observe that QLoRA consistently has the lowest memory footprint because the pre-trained weights are represented in lower precision. LoRA exhibits higher throughput leveraging the optimization in LoRA layers by Unsloth. GaLore achieves lower PPL on large models while LoRA advantages on smaller ones.

5.2 Fine-Tuning on Downstream Tasks

Experimental Setup To evaluate the effectiveness of different efficient fine-tuning methods, we compare the performance of various models after fine-tuning on downstream tasks. We construct non-overlapping training set and test set using 2,000 examples and 1,000 examples from three representative text generation tasks, including CNN/DM (Nallapati et al., 2016), XSum (Narayan et al., 2018) and AdGen (Shao et al., 2019), respectively. We select several instruction-tuned models and fine-tune them following the sequence-to-sequence task using different fine-tuning methods. Then we compare the results of full-tuning (FT), GaLore, LoRA and 4-bit QLoRA. After fine-tuning, we calculate the ROUGE score (Lin, 2004) on the test set of each task. We also incorporate the scores of the original instruction-tuned models as baselines.

Model	CNN / DM					XSum					AdGen				
	Baseline	FT	GaLore	LoRA	QLoRA	Baseline	FT	GaLore	LoRA	QLoRA	Baseline	FT	GaLore	LoRA	QLoRA
ChatGLM3-6B	18.51	22.00	22.16	21.68	21.70	16.14	26.25	26.34	26.50	26.78	14.53	19.91	20.57	20.47	20.49
Yi-6B	16.85	22.40	22.68	22.98	22.97	18.24	27.09	28.25	28.71	29.21	13.34	19.68	20.06	20.97	20.31
Llama2-7B	12.94	22.87	22.40	22.70	22.61	13.89	27.69	27.64	28.80	28.05	0.61	20.51	19.61	20.29	20.45
Mistral-7B	14.39	22.03	22.99	23.47	23.28	15.87	23.57	28.00	30.41	30.44	7.82	20.14	20.90	20.99	20.56
Gemma-7B	15.97	22.07	/	22.41	22.44	15.31	25.13	/	28.67	29.02	11.57	19.99	/	20.62	19.81
Qwen1.5-7B	15.40	22.46	21.76	22.71	22.52	19.27	26.68	26.64	27.77	27.60	14.49	20.42	21.08	21.31	21.34
Qwen2-7B	16.46	23.20	/	23.29	23.66	19.76	26.94	/	28.92	28.94	12.89	19.83	/	20.96	20.86
Llama3-8B	15.19	23.36	23.57	23.48	**24.12**	17.83	26.21	30.45	30.63	**30.94**	0.22	20.28	21.27	**21.44**	21.20

Table 5: Comparison of the performance (in terms of ROUGE) on specific tasks using different fine-tuning methods in LLAMAFACTORY. The best result of each model is underlined, and the best result of each task is in **bold**.

In this experiment, we set learning rate to 10^{-5}, batch size to 4 and maximum input length to 2048. We fine-tune these models using the 8-bit AdamW optimizer (Dettmers et al., 2022b) in bfloat16 precision with activation checkpointing. For GaLore, we set the rank and scale to 128 and 2.0, respectively. For LoRA and QLoRA, we attach adapters to all linear layers and set the rank and alpha to 128 and 256, respectively. All the experiments are conducted on NVIDIA A100 40GB GPUs.

Results The evaluation results on downstream tasks are shown in Table 5. We report the averaged scores over ROUGE-1, ROUGE-2 and ROUGE-L. Some results of the Gemma-7B and Qwen2-7B (Bai et al., 2023) models are not included in the table because the GaLore method may not be applicable to them. An interesting finding from the results is that LoRA and QLoRA achieve the best performance in most cases, except for the ChatGLM3-6B (Zeng et al., 2024) and Llama2-7B models on the CNN/DM and AdGen datasets. This phenomenon highlights the effectiveness of these efficient fine-tuning methods in adapting LLMs to specific tasks. Additionally, we observe that Llama3-8B achieves the best performance among these models, while Yi-6B (Young et al., 2024) and Mistral-7B (Jiang et al., 2023a) exhibit competitive performance among models of the same size.

6 Conclusion and Future Work

In this paper, we demonstrate LLAMAFACTORY, a unified framework for the efficient fine-tuning of LLMs. Through a modular design, we minimize dependencies between the models, datasets and training methods and provide an integrated approach to fine-tune over 100 LLMs with a diverse range of efficient fine-tuning techniques. Additionally, we offer a flexible web UI LLAMABOARD, enabling customized fine-tuning and evaluation of LLMs without coding efforts. We empirically validate the efficiency and effectiveness of our framework on language modeling and text generation tasks.

We will consistently keep LLAMAFACTORY synchronous with the state-of-the-art models and efficient fine-tuning techniques. We also welcome contributions from the open-source community. The road map of LLAMAFACTORY including:

(1) Enabling fine-tuning for models that supports a wider range of modalities, *e.g.*, the audio and video modalities (Zhu et al., 2024).

(2) Integrating more parallel training strategies, *e.g.*, sequence parallelism (Jacobs et al., 2023) and tensor parallelism (Shoeybi et al., 2019).

(3) Exploring stronger fine-tuning methods for conversational models, *e.g.*, self-play (Chen et al., 2024b; Yuan et al., 2024).

7 Broader Impact and Responsible Use

LLAMAFACTORY has attracted a large number of individuals interested in LLMs to explore the possibility of customizing models. This contributes significantly to the growth of the open-source communities. It is gaining increasing attention and is being featured in Awesome Transformers[2] as a representative of efficient fine-tuning frameworks for LLMs. We anticipate that practitioners build their LLMs upon our framework that bring benefits to society. Adherence to the model license is mandatory when using LLAMAFACTORY for fine-tuning LLMs, thus preventing from any potential misuse.

Acknowledgements

This work is supported partly by the National Science and Technology Major Project under Grant 2022ZD0120202, by the National Natural Science Foundation of China (No. U23B2056), by the Fundamental Research Funds for the Central Universities, and by the State Key Laboratory of Complex & Critical Software Environment.

[2]https://github.com/huggingface/transformers/blob/v4.40.0/awesome-transformers.md#llama-factory

References

Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ml models in the wild. In *ICML Workshop on Human in the Loop Learning*, Long Beach, USA.

Lightning AI. 2023. Lit-gpt.

Yuvanesh Anand, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt, and Andriy Mulyar. 2023. GPT4All: Training an assistant-style chatbot with large scale data distillation from GPT-3.5-turbo.

Apache. 2016. Arrow.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open LLM leaderboard.

Rishabh Bhardwaj, Do Duc Anh, and Soujanya Poria. 2024. Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic. *arXiv preprint arXiv:2402.11746*.

Kathi Canese and Sarah Weis. 2013. PubMed: the bibliographic database. *The NCBI handbook*, 2(1).

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024a. LongLoRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tim Dettmers. 2021. Bitsandbytes.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022a. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022b. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36:10088–10115.

Shizhe Diao, Rui Pan, Hanze Dong, KaShun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2024. LMFlow: An extensible toolkit for finetuning and inference of large foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 116–127, Mexico City, Mexico. Association for Computational Linguistics.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. In *International Conference on Machine Learning*, Vienna, Austria. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*.

Daniel Han and Michael Han. 2023. unsloth.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning*, Vienna, Austria. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. 2023. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Leon Song, Samyam Rajbhandari, and Yuxiong He. 2023. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2023b. ReasoningLM: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3721–3735, Singapore. Association for Computational Linguistics.

Wenxiang Jiao, Jen-tse Huang, Wenxuan Wang, Zhiwei He, Tian Liang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023a. ParroT: Translating during chat using large language models tuned with human translation and feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15009–15020, Singapore. Association for Computational Linguistics.

Yizhu Jiao, Ming Zhong, Sha Li, Ruining Zhao, Siru Ouyang, Heng Ji, and Jiawei Han. 2023b. Instruct and extract: Instruction tuning for on-demand information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10030–10051, Singapore. Association for Computational Linguistics.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with LoRA. *arXiv preprint arXiv:2312.03732*.

Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. 2021. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023a. CMMLU: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. 2023b. Colossal-AI: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 766–775.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, Vienna, Austria. PMLR.

Qijun Luo, Hengxu Yu, and Xiao Li. 2024. BAdam: A memory efficient full parameter training method for large language models. *arXiv preprint arXiv:2404.02827*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI blog*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 37.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-offload: Democratizing billion-scale model training. In *USENIX Annual Technical Conference*, pages 551–564.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3257–3268, Hong Kong, China. Association for Computational Linguistics.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Sang Truong, Duc Nguyen, Toan Nguyen, Dong Le, Nhi Truong, Tho Quan, and Sanmi Koyejo. 2024. Crossing linguistic horizons: Finetuning and comprehensive evaluation of Vietnamese large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2849–2900, Mexico City, Mexico. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. TRL: Transformer reinforcement learning.

Chenglong Wang, Hang Zhou, Yimin Hu, Yifu Huo, Bei Li, Tongran Liu, Tong Xiao, and Jingbo Zhu. 2023a. ESRL: Efficient sampling-based reinforcement learning for sequence generation. *arXiv preprint arXiv:2308.02223*.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023b.

Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore. Association for Computational Linguistics.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023c. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01.ai. *arXiv preprint arXiv:2403.04652*.

Hao Yu, Zachary Yang, Kellin Pelrine, Jean Francois Godbout, and Reihaneh Rabbany. 2023. Open, closed, or small language models for text classification? *arXiv preprint arXiv:2308.10092*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, et al. 2024. ChatGLM: A family of large language models from GLM-130b to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*.

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2024. LLaMA-adapter: Efficient fine-tuning of language models with zero-init attention. In *International Conference on Learning Representations*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. GaLore: Memory-efficient llm training by gradient low-rank projection. In *International Conference on Machine Learning*, Vienna, Austria. PMLR.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, WANG HongFa, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. 2024. LanguageBind: Extending video-language pretraining to N-modality by language-based semantic alignment. In *International Conference on Learning Representations*.