Large Language Model Fine-tuning with Low-Rank Adaptation: A Performance Exploration

Bagus Hanindhito
hanindhito@bagus.my.id
The University of Texas at Austin
Austin, Texas, USA

Bhavesh Patel
Bhavesh.A.Patel@dell.com
Dell Technologies
Round Rock, Texas, USA

Lizy K. John
ljohn@ece.utexas.edu
The University of Texas at Austin
Austin, Texas, USA

Abstract

Fine-tuning pre-trained models is the preferred method for adapting large language models (LLMs) for specific downstream tasks since it is significantly more efficient in terms of computational costs and energy than training the models from scratch. However, with LLMs experiencing exponential growth, fine-tuning the models becomes more challenging and expensive as they demand more computational resources. Many approaches are proposed to fine-tune state-of-the-art models efficiently, reducing the infrastructure needed, and thus, making them accessible to the public.

In this paper, we investigate a technique called Low-Rank Adaptation (LoRA), one approach to efficiently fine-tuning LLMs by leveraging low intrinsic dimensions possessed by the models during fine-tuning. Specifically, we explore different data formats that can be used during LoRA fine-tuning and compare them regarding workload performance and model accuracy. The experiment compared LoRA and its quantized counterpart (QLoRA) with regular methods to fine-tune state-of-the-art LLMs. The analysis includes estimating memory usage, measuring resource utilization, and evaluating the model quality after fine-tuning. Three state-of-the-art Graphics Processing Units (GPUs) are used for experiments, including NVIDIA H100, NVIDIA A100, and NVIDIA L40. We also use the newest AMD MI300X GPU as a preliminary exploration.

The experiment shows that although LoRA with a 16-bit floating-point format can significantly reduce the computational resource demand, it still requires data-center-class GPUs with ample memory to fine-tune LLMs with 70 billion parameters. Using QLoRA with 4-bit floating-point format significantly lowers the memory requirements by as much as 75% compared to LoRA, allowing a single GPU with 48 GB and 80 GB of memory to fine-tune 70 billion parameter models. In addition, QLoRA delivers model quality that is on par with or exceeds the quality of the model obtained from conventional fine-tuning.

CCS Concepts

• **General and reference** → **Evaluation**; **Performance**; **Measurement**; **Experimentation**; • **Computing methodologies** → *Machine learning*; *Natural language generation.*

ACM ISBN 979-8-4007-1073-5/2025/05
https://doi.org/10.1145/3676151.3719377

Keywords

Performance Exploration, Large Language Models, Low-Rank Adaptation, Data Formats, Fine-Tuning

ACM Reference Format:
Bagus Hanindhito, Bhavesh Patel, and Lizy K. John. 2025. Large Language Model Fine-tuning with Low-Rank Adaptation: A Performance Exploration. In *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE '25), May 5–9, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3676151.3719377

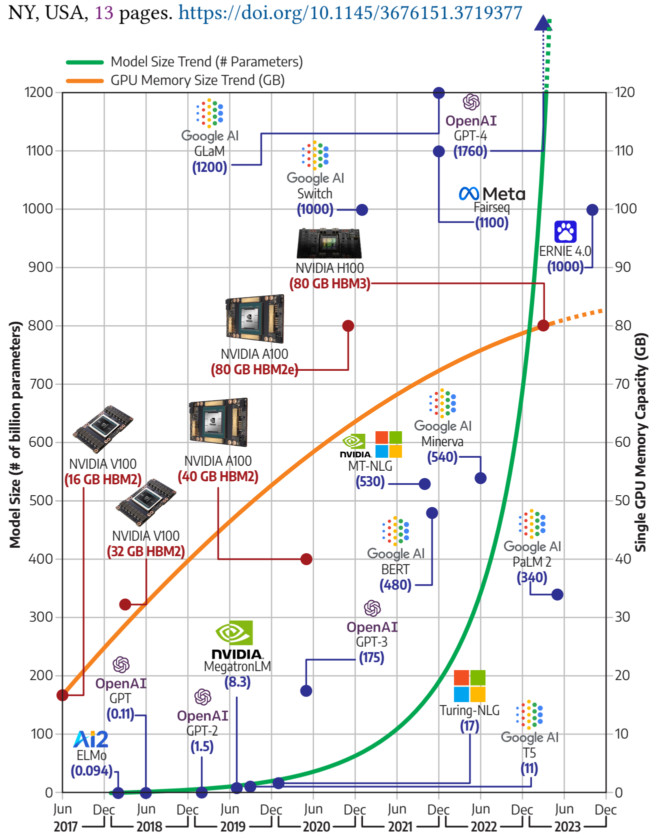

Figure 1: The imbalance trend between the growth of large language model size (green line, in billion parameters) and the increase in GPU memory capacity (orange line).

1 Introduction

Large Language Models (LLMs) have been growing exponentially following the neural network scaling laws [13, 35, 39], gaining popularity in recent years [36, 37, 44, 46, 58, 73, 89]. LLMs size grew by a factor of 1000× between 2018 and 2020, while during the same period, the memory capacity of Graphics Processing Units

(GPUs), popular accelerators for machine learning [32, 43, 57, 59, 72, 91], only saw 5× increase [60, 61]. Single GPU is no longer sufficient to train state-of-the-art LLMs; hundreds or thousands of GPUs are needed, making training LLMs more costly [16, 80], and significantly impacting the environment [5, 12, 70, 83].

With the increasing costs of training LLMs from scratch, fine-tuning is the preferred method for adapting LLMs to perform specific downstream tasks [20, 28, 78]. This involves taking available pre-trained models and subjecting them to more particular datasets. However, as the size of LLMs grows exponentially, even fine-tuning the models becomes prohibitively expensive, necessitating finding more efficient methods. One approach to efficiently fine-tune LLMs is Low-Rank Adaptation (LoRA), introduced in 2022 by researchers at Microsoft [40]. Further improvement of LoRA comes from using smaller data formats through quantization to reduce memory requirements, as seen with Quantized LoRA (QLoRA), introduced by the researcher at the University of Washington in 2023 [18].

This paper explores the fine-tuning performance under different GPU architectures and the model's performance under different data formats on conventional fine-tuning, LoRA fine-tuning, and QLoRA fine-tuning for state-of-the-art LLMs, including Llama [86], Llama2 [87], Falcon [3], and WizardLM [98]. The experiments done in this paper are unique since they involve performance measurements on state-of-the-art Graphics Processing Units (GPUs) at the time of writing, including NVIDIA H100, NVIDIA A100, and NVIDIA L40. We also use the newest AMD MI300X GPU as preliminary performance exploration, making our paper among the first to investigate the performance of this GPU to fine-tune LLMs using LoRA and QLoRA. Specifically, the objectives of our paper are the following:

- We briefly summarize the LoRA and QLoRA fine-tuning methods compared to conventional fine-tuning methods to familiarize them with general readers (Sections 2.4 and 2.5).
- We perform fine-tuning of the latest state-of-the-art LLMs: Llama, Llama2, Falcon, and WizardLM with different numbers of parameters on three different GPUs: NVIDIA A100 80 GB, NVIDIA H100 80 GB, and NVIDIA L40 48 GB (Section 4.2).
- We estimate the memory required for fine-tuning the LLMs and correlate it to the actual memory usage (Sections 4.1 and 4.2.3).
- We measure the time needed to fine-tune the LLMs using conventional, LoRA, and QLoRA methods on different computation data formats: FP32, BF16, and FP16. In addition, we also compare the effect of quantization on QLoRA for two data formats, NF4 and FP4, as well as multi-level quantization (Sections 4.2.1 and 4.2.2).
- We evaluate the quality of fine-tuned LLMs using Massive Multitask Language Understanding (MMLU) benchmark [34], which then is used to compare LoRA and QLoRA against conventional fine-tuning flow. We also analyze the effect of quantization on the model quality of QLoRA for NF4 and FP4 data formats and multi-level quantization (Section 4.3).
- We perform early performance exploration with AMD MI300X GPU and investigate the behavior of the software stack when running LoRA and QLoRA for fine-tuning LLM (Section 4.4).

The major insights of this paper are summarized as follows:

- For small-size LLMs (i.e., 7 billion parameters or less), NVIDIA H100 and A100 with 80 GB memory are sufficient to perform conventional fine-tuning by leveraging paged optimizers. While excessive data movement between CPU and GPU degrades the overall performance, consumer-grade GPUs with lower memory capacity cannot be used to fine-tune these models.
- For large-size LLMs (i.e., 40 billion parameters or more), single GPU available today, even with 192 GB of GPU memory on AMD MI300X, is no longer viable; it needs multi-GPU setup.
- LoRA significantly reduces the memory utilization, allowing GPU with less than 48 GB of memory to fine-tune small-size LLMs (i.e., 7 billion parameters or less) at the expense of slightly longer fine-tuning time compared to conventional fine-tuning due to overhead associated with LoRA. However, LoRA is no longer sufficient to fine-tune large-size LLMs (i.e., 40 billion parameters or more) using single GPU available today.
- QLoRA further reduces the memory utilization of LoRA by as much as 75%, allowing single GPU with 80 GB memory to fine-tune larger-size LLMs (i.e., 40 billion parameters or more) at the expense of more computation overhead due to quantization.
- Models fine-tuned with LoRA or QLoRA give on-par or better accuracy than standard fine-tuning. Specifically for QLoRA, the NF4 provides better accuracy compared to FP4.
- While AMD MI300X provides the highest memory capacity and the highest number of vector units at the time of writing, the software stack needs to be further optimized and refined to get the most performance out of the hardware. Relying on the compiler to port available codes is not sufficient.

2 Background

2.1 Hardware and ML Workload Trend

The models' size and the dataset to train them are growing exponentially as they follow the neural network scaling laws [13, 35, 39]. Obtaining higher accuracy models can often be accomplished by increasing the size of the models [10, 74] and exposing them to the vast amount of high-quality datasets during the training [4, 7, 37]. This is especially true for the recently-popular LLMs [19, 29, 36–38, 44, 46, 58, 67, 73, 89] that find their ways into many applications [8, 9, 21, 22, 33, 41, 52, 66, 81, 84, 85, 88, 94, 95, 97, 99, 100, 102–104, 107]. Between 2018 and 2020, the size of LLMs increased by a factor of 1000: from 94 million parameters ELMo introduced in 2018 [71] to 175 billion parameters GPT-3 introduced in 2020 [11]. The introduction of ChatGPT at the end of 2022 [26, 77, 96] marked the beginning of the Generative AI era [6, 23, 24, 45], which demands even larger models [14, 24]. Its successor, GPT-4, was released in March 2023 and is estimated to have 1.76 trillion parameters [69].

In contrast, within the same 2-year period, Graphics Processing Units (GPUs), the popular accelerators for training AI and ML [31, 32, 43, 57, 59, 72, 91], only see a 5× increase in memory capacity: from NVIDIA Tesla V100 with 16 GB of HBM2 memory released in June 2017 [60] to the NVIDIA A100 with 80 GB of HBM2E memory released in November 2020 [61]. Since its successor, the NVIDIA H100 [62], still retains the same 80 GB memory capacity, we need more than three years to see GPUs with double that memory capacity: AMD MI300X with 192 GB of HBM3 memory released in December 2023 [1], NVIDIA H200 with 141 GB of HBM3E memory released in the second quarter of 2024 [65], and the upcoming NVIDIA B100 and B200 GPUs with 192 GB of HBM3E memory expected to be launched at the second half of 2024 [64].

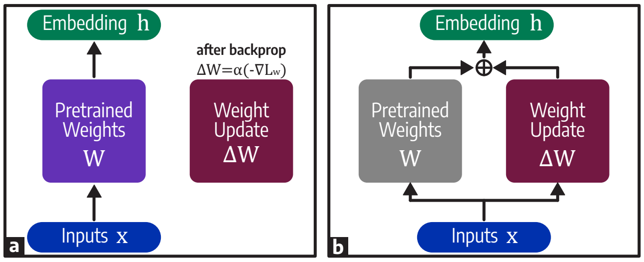

Figure 2: Conventional fine-tuning flow (a) and its alternative counterpart (b).

Due to the trend imbalance shown in Figure 1, hundreds or even thousands of GPUs are required to handle state-of-the-art LLMs by aggregating computational power, memory, and bandwidth [42, 79, 82, 105]. This requires building expensive infrastructure, making training models from scratch more expensive [16, 80]. For example, training GPT-3 and GPT-4 models could cost more than $5M [53] and $100M [48], respectively, as estimated from the infrastructure required to handle such models. In addition, training such models has significant environmental impacts due to the enormous energy consumed [5, 12, 70, 83].

2.2 Conventional Model Fine-tuning

Fine-tuning becomes the favored method for adopting the models for specific downstream tasks since training the models from scratch is prohibitively expensive. In fine-tuning, one can take pre-trained models trained from scratch using more general datasets and subject them to more specific datasets to adapt them to new specific tasks (i.e., downstream tasks). In addition to saving significant time, computational resources, and energy, using pre-trained models for fine-tuning has the benefit of generalization and regularization, reducing overfitting and improving the fine-tuned model's performance and accuracy for downstream tasks [25, 54, 101]. Fine-tuning also does not require a huge amount of data, which is beneficial for tasks where the dataset is small and scarce [55].

Figure 2 (a) illustrates the high-level overview of conventional fine-tuning of a pre-trained LLM. The W, which is the pre-trained model's weight, is subjected to short training on the specific datasets tailored for the target downstream tasks. After backpropagation, the weight updates, ΔW, are obtained by multiplying the negative gradient of the loss, $-\nabla L_W$, and the learning rate α. The weight updates, ΔW, are used to update the pre-trained model's weight W, which is then used to generate the output h. Alternatively, ΔW and W can be stored as separate matrices as shown in Figure 2 (b) where the W is frozen (i.e., not changed or updated) after the fine-tuning. The output, h, can be computed using $h = Wx + \Delta Wx$. While this means that it needs double the memory to store both W and ΔW separately, its benefit will become more apparent when Low-Rank Adaptation (LoRA) is introduced (Section 2.4).

Although conventional fine-tuning previously promised a more economical and efficient way of adapting LLMs, it has become more demanding as the LLMs become larger. In addition to the pre-trained weights, the optimizer states and gradients consume a significant amount of memory, which easily exceeds the memory capacity of single GPU (Section 4.1). Multi-GPU systems are needed to fine-tune state-of-the-art LLMs, necessitating the finding of more efficient fine-tuning methods.

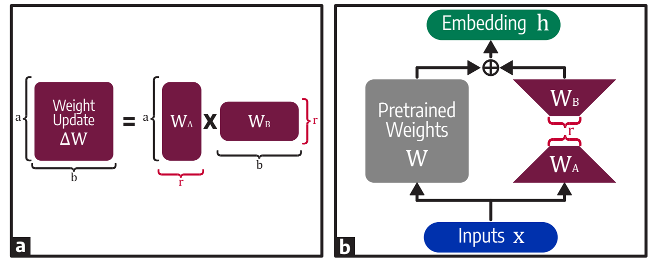

Figure 3: LoRA fine-tuning is performed by decomposing the weight update matrix into two lower-rank matrices (a), which are then used for fine-tuning (b).

2.3 Intrinsic Dimensionality of Models

In their work published in 2020, Aghajanyan et al. analyzed the behavior of the models during the fine-tuning using intrinsic dimensions [2]. Their goal is to find the minimum number of free parameters required to closely approximate the quality of the models when full-parameter fine-tuning is used. In other words, instead of using the whole pre-trained weights during fine-tuning as shown in Figure 2, their objective is to find the smaller representation of the model for fine-tuning without losing too much information. Their investigation showed that pre-trained models have significantly fewer intrinsic dimensions, and thus, there exists a lower dimension representation of the models that are as effective as their full parameter counterparts for fine-tuning.

2.4 Lower Rank Representation of Models

Based on the finding summarized in Section 2.3, an efficient fine-tuning method called Low-Rank Adaptation (LoRA) was proposed by Hu et al. from Microsoft in 2021 [40]. Leveraging the fact that models can have lower intrinsic dimensions during fine-tuning, lower-dimension matrices can replace the weight updates matrix, ΔW, without losing too much information. As shown in Figure 3 (a), the ΔW matrix with dimension $a \times b$ can be decomposed into two smaller rank matrices, W_A and W_B, whose dimensions are $a \times r$ and $r \times b$, respectively. With these two matrices, the LoRA fine-tuning is performed as shown in Figure 3 (b).

Replacing the ΔW matrix with two smaller LoRA matrices, W_A and W_B, dramatically reduces the number of trainable parameters. For example, replacing ΔW whose dimension is 1000×1000 with two matrices whose dimensions are 1000×5 and 5×1000 reduces the number of trainable parameters by 99% (i.e., 1 million vs. 10,000 parameters). The lower number of trainable parameters means that the number of gradients and optimizer states is significantly reduced, greatly reducing the memory requirements. The W matrix in its original dimension is frozen (i.e., not updated during fine-tuning), and thus, it does not require optimizer states and gradients.

The matrix's rank, r, becomes the LoRA hyperparameter that controls the size of the LoRA matrices. A smaller r implies fewer trainable parameters, resulting in faster fine-tuning and lower required compute resources, but at the expense of a reduced model's ability to capture task-specific information. Therefore, by adjusting r, we can control the trade-off between model complexity, model adaptation ability, and the cost of fine-tuning.

Finally, we would like to highlight the benefit of having separate matrices to store pre-trained weight W and weight update

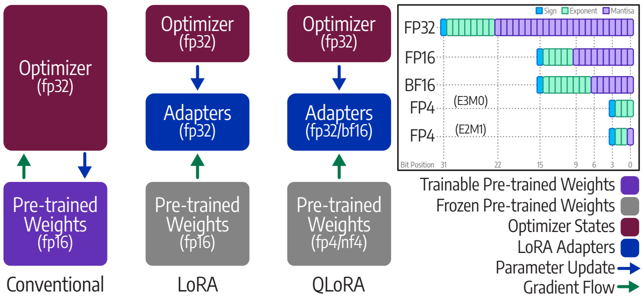

Figure 4: Comparison of conventional, LoRA, and QLoRA fine-tuning in terms of data format usage. Note that there is no standardized FP4 format.

ΔW, as briefly mentioned in Section 2.2. With multiple fine-tuned models derived from the same pre-trained model, one can store one W matrix and numerous pairs of LoRA matrices W_A and W_B corresponding to each fine-tuned model, which is significantly smaller than the W matrix. This saves a tremendous amount of storage/memory compared to storing the whole dimension of updated W for each fine-tuned model. This is why these two LoRA matrices are also called LoRA adapters.

2.5 Quantization for Low-Rank Adaptation

Although LoRA promises to significantly reduce the memory requirement for fine-tuning LLMs, the W matrix can still be huge for large models. For example, LoRA still needs 100 GB of GPU memory to fine-tune the 65-billion-parameter 16-bit Llama model (Section 4.1). Although the memory requirement is already reduced significantly compared to 780 GB needed in conventional fine-tuning, it is still beyond the capacity of single data-center class GPU (e.g., 80 GB NVIDIA A100 or 80 GB NVIDIA H100), let alone the consumer class GPU that usually has lower memory capacity.

In 2023, Dettmers et al. from the University of Washington proposed an improvement to LoRA called Quantized LoRA (QLoRA) [18]. In summary, QLoRA stores the W matrix in quantized 4-bit floating-point formats, significantly reducing the memory required. Figure 4 compares conventional, LoRA, and QLoRA fine-tuning regarding data format usage. Three significant improvements of QLoRA compared to LoRA are explained as follows.

2.5.1 FP4 and NF4 Quantization. Unlike LoRA, which stores the W matrix in a 16-bit floating-point format (FP16), QLoRA stores it in a 4-bit floating-point format through quantization. QLoRA supports two 4-bit floating-point formats: FP4 and NF4. For FP4, there is no standardized fixed format; it can be E3M0 to prioritize dynamic range, E2M1 to get more accuracy, and E1M2 to prioritize accuracy. The E3M0 often performs better due to the larger dynamic range. On the other hand, the NF4 format stands for normal-float, which is an information-theoretically optimal data format. The NF4 format is obtained through quantization using an empirical cumulative distribution function where each quantization bin has an equal number of values based on W. The authors claim that NF4 gives better fine-tuning quality than FP4 and 4-bit Integer (INT4) formats.

While the LoRA adapters are stored in 32-bit floating-point format (FP32), QLoRA allows storing its adapters either in FP32 or a special 16-bit floating-point format called BF16 [92], which retains the same dynamic range as FP32 while sacrificing precision. Using BF16 on hardware that has native support for it, such as Tensor Cores on NVIDIA A100 [61], NVIDIA H100 [62], and NVIDIA L40 [63] or Matrix Core on AMD MI300X [1] GPUs can reduce memory requirements and improve computation performance.

It is important to note that although the W is quantized and stored in 4-bit floating-point format (FP4/NF4), the fine-tuning is still done in either mixed precision (FP32/FP16) or BF16 to preserve accuracy. This requires dequantization of the pre-trained weights before they can be used for computation, which may add additional compute overhead.

2.5.2 Double Quantization. In addition to the quantized values, the 4-bit quantization of W results in quantization constants, which are the overhead of quantization. In QLoRA, quantizing 64 values results in a 32-bit floating-point (FP32) quantization constant, which gives an overhead of 0.5 bits per model parameter. To further lower the quantization overhead, QLoRA uses second-level quantization, which quantizes the quantization constants. A group of 256 first-level quantization constants is quantized, resulting in one 8-bit floating-point (FP8) second-level quantization constant. This reduces the quantization overhead to 0.127 bits per model parameter, translating to 3 GB memory saving on Llama with 65 billion parameters.

2.5.3 Paged Adam Optimizer. The Paged Adam optimizer allows QLoRA to utilize NVIDIA Unified Virtual Memory (UVM) to store the optimizer states in GPU and CPU memory. In the case of insufficient GPU memory to store the optimizer states, CPU memory is used to store parts of them, and the NVIDIA UVM takes care of the data movement between GPU memory and CPU memory. However, significant spillage will quickly degrade overall performance due to excessive data movement between CPU and GPU through the PCI Express bus.

3 Methods

3.1 Hardware and Software Setup

The experiments are performed on two different compute platforms: Dell PowerEdge XE9680 and Dell PowerEdge R760xa. With identical CPU configuration, the Dell PowerEdge XE9680 platform has three different GPU configurations: eight AMD MI300X GPUs (`MI300X`) [1], eight NVIDIA H100 GPUs (`H100`) [62], and eight NVIDIA A100 GPUs (`A100`) [61]. On the other hand, only one GPU configuration for R760xa: four NVIDIA L40 GPUs (`L40`) [63]. Table 1 summarizes the platform configurations. The vector unit is called CUDA Cores and Stream Processors in NVIDIA and AMD GPUs, respectively, while the matrix unit is called Tensor Cores and Matrix Cores in NVIDIA and AMD GPUs, respectively.

Configurations that use NVIDIA GPUs are equipped with CUDA 12.2 and NVIDIA driver 535.86.10. To leverage the newer CUDA 12.2, PyTorch [68] version 2.2.0 is built from scratch inside an Anaconda 23.7.2 environment. On the other hand, configurations that use AMD GPUs are equipped with Radeon Open Compute (ROCm) 6.0, AMD driver 6.7.0, and officially-built PyTorch version 2.3.0 for the ROCm Platform.

Table 1: Hardware Configuration

Platform	**XE9680**			**R760xa**
GPU				
Manufacturer	**AMD**	**NVIDIA**		
Model (# GPUs)	MI300X (8)	H100 (8)	A100 (8)	L40 (4)
Form Factor	OAM	SXM5	SXM4	PCIe
# Vector Unit	19456	16896	6912	18176
# Matrix Unit	1216	528	432	568
Memory Size	192 GB	80 GB	80 GB	48 GB
Memory Type	HBM3	HBM3	HBM2E	GDDR6
Bandwidth	5427 GBps	3350 GBps	2039 GBps	864 GBps
Typical Power	750 W	700 W	500 W	300 W
CPU (2 Sockets)				
Model	Xeon 8470			Xeon 6430
Base Clock	2.00 GHz			2.10 GHz
# Total Cores	104			64
Memory Size	2048 GB			512 GB
Memory Type	DDR5-4400			DDR5-4400
Bandwidth	281 GBps			281 GBps
Typical Power	350 W			270 W
Interfaces				
CPU-to-CPU	UPI 16 GT/s			
CPU-to-GPU	PCIe 5.0 x16		PCIe 4.0 x16	
GPU-to-GPU	∞ Fabric 4.0	NVLink 4.0	NVLink 3.0	None

Table 2: LLMs in Experiment

Name	Developer	# Parameters	HuggingFace Hub Link
Llama	Meta AI	7 billion	huggyllama/llama-7b
Llama	Meta AI	65 billion	huggyllama/llama-65b
Llama2	Meta AI	7 billion	meta-llama/Llama-2-7b-hf
Llama2	Meta AI	70 billion	meta-llama/Llama-2-70b-hf
Falcon	TII UAE	7 billion	tiiuae/falcon-7b
Falcon	TII UAE	40 billion	tiiuae/falcon-40b
WizardLM	Microsoft	7 billion	WizardLM/WizardLM-7B-V1.0
WizardLM	Microsoft	70 billion	WizardLM/WizardLM-70B-V1.0

In addition, several libraries are used for the experiments. While most of the libraries are written to give more optimized performance for NVIDIA GPUs, they may not be optimized for AMD GPUs. Some of the libraries need to be built from sources, relying on the compiler provided by ROCm to port the codes from NVIDIA to AMD GPUs. The following is the list of third-party libraries.

- HuggingFace Transformers [93], which provides APIs for quick interaction with pre-trained models.
- HuggingFace Evaluate [90], which provides tools for evaluating and comparing models' performance.
- HuggingFace PEFT [56], which provides access to state-of-the-art parameter efficient fine-tuning, including LoRA.
- HuggingFace Accelerate [27], which provides an abstraction to run PyTorch in any device, including multi-GPU.
- DeepSpeed ZeRO [75, 76], which provides library for distributed training on multi-GPU.
- BitsandBytes [17], which provides 4-bit and 8-bit quantization for pre-trained weights for QLoRA.

Due to limited space, we provide a more detailed experimental setup in an open repository accessible through Zenodo [30]. The repository contains scripts, guidance, and log files for interested readers to be able to replicate the experiments done in this paper.

3.2 Model, Dataset, and Evaluation

The experiments use four popular large language models as summarized by Table 2: the Llama (`Llama-7B`, `Llama-65B`) [86] and its successor, Llama2 (`Llama2-7B`, `Llama2-70B`) [87] from Meta AI, the Falcon (`Falcon-7B`, `Falcon-40B`) [3] from Technology Innovation Institute of UAE, and the WizardLM (`WizardLM-7B`, `WizardLM-70B`) from Microsoft and Peking University [98]. The experiment will mostly focus on handling the smallest variant of each model (i.e., 7 billion parameters) with limited discussion on the largest variant (i.e., 65-billion-parameter Llama, 70-billion-parameter Llama2, 40-billion-parameter Falcon, and 70-billion-parameter WizardLM).

In general, the experiments compare conventional fine-tuning (`std`) with LoRA (`LoRA`) and QLoRA (`QLoRA`) fine-tuning in terms of fine-tuning performance (i.e., the time needed to fine-tune the models), resource usage (i.e., CPU memory, GPU memory), and model quality. The dataset used for fine-tuning the models is the OpenAssistant Conversation Dataset (OASST1) [50], which can also be downloaded from HuggingFace Hub (OpenAssistant/oasst1). Finally, the model quality is measured using the Massive Multitask Language Understanding (MMLU) benchmark [34] after 2048 steps of the fine-tuning process. For QLoRA, additional experiments are performed to observe the effects of FP4 and NF4 data formats on model quality, the impact of double quantization on memory usage and model quality, and the performance advantages of using BF16 instead of mixed precision FP32/FP16.

4 Evaluation and Discussion

4.1 Estimating Memory Requirements

The number of parameters is roughly used to estimate the memory requirements. In conventional fine-tuning (`std`), all parameters are trainable, and each needs optimizer states and gradients. On the other hand, `LoRA` and `QLoRA` freeze the model parameters (Figure 4) and add the adapters. Only the adapters have trainable parameters, reducing the memory due to fewer optimizer states and gradients.

In PyTorch, parameters are trainable when `requires_grad` property is set to `True`. Therefore, we can determine the total number of trainable parameters of the models and adapters using this property. Note that, for `LoRA` and `QLoRA`, the calculation must be done after the adapters are attached and pre-trained weights are frozen. Special handling is required for `QLoRA`, which uses 4-bit quantization. Computer memory is byte-addressable, so the smallest unit is a byte. To store parameters in the memory, two 4-bit values must be packed together to form a byte, which is taken care of by the BitsandBytes library [17]. In other words, one byte contains two model parameters for `QLoRA` with 4-bit quantization.

The next step is to approximate memory usage based on the number of parameters (trainable and frozen) and the data formats they use. Property `dtype` can be used for each variable in PyTorch to determine the data formats for model parameters and adapters. Note that the data format shown for `QLoRA` with 4-bit quantization for pre-trained weight is an 8-bit integer for the reason explained in the previous paragraph. In addition to storing the pre-trained weights in the memory, each trainable parameter will need optimizer states and gradients in a 32-bit floating-point format (FP32). For Adam Optimizer [47], each trainable parameter needs two state variables, and hence, 8 bytes (i.e., two FP32 values) per trainable parameter. For gradients, 4 bytes are required per trainable parameter.

Additional memory required and not included in our approximation includes memory for forward activations, datasets, quantization constants in `QLoRA`, temporary variables, and unusable memory

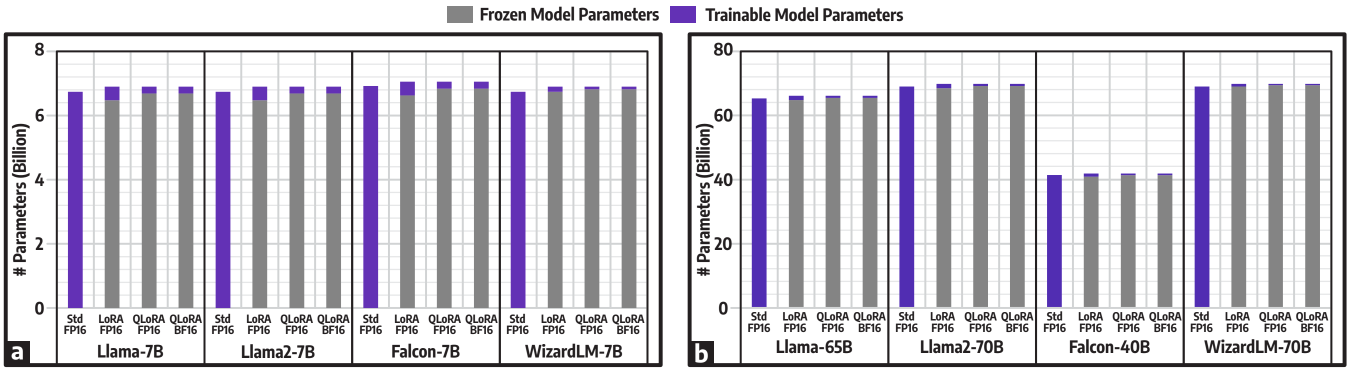

Figure 5: Comparison of conventional, LoRA, and QLoRA fine-tuning in terms of the number of trainable parameters for small-size models (a) and large-size models (b). In LoRA and QLoRA, the trainable parameters come from the adapters, while the pre-trained weights are frozen.

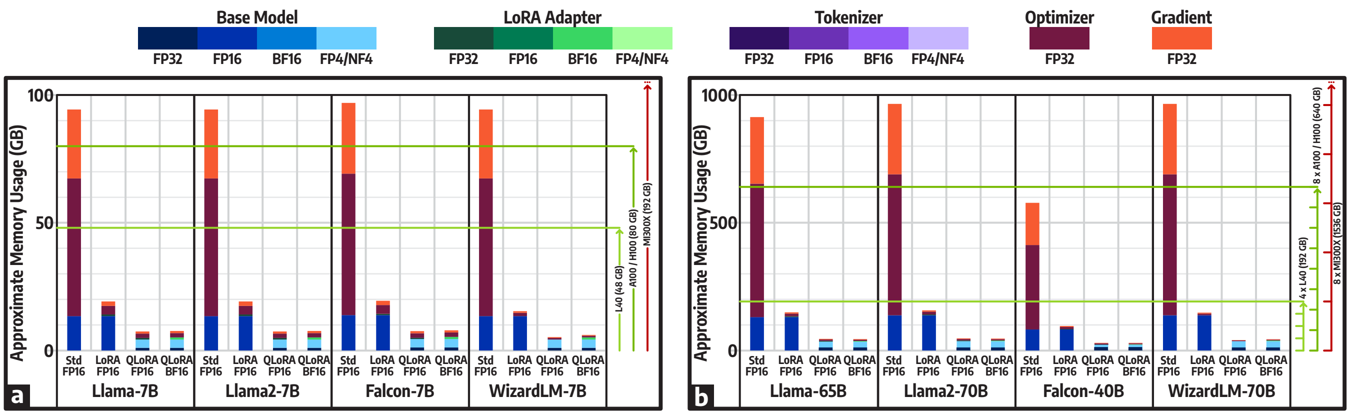

Figure 6: Approximate memory usage for conventional, LoRA, and QLoRA fine-tuning of small-size models (a) and large-size models (b). For small-size models, single GPU memory capacity is shown on the right for reference, while for large-size models, aggregate GPU memory capacity inside the compute platform is shown instead.

due to fragmentation. The memory size for forward activations depends on the model configuration, which includes the sequence length and hidden size. Therefore, the actual memory may be larger than the approximation derived in this section.

4.1.1 Memory for Small-Size Models. Figure 5 (a) shows the trainable (purple) and frozen (grey) parameters for `Llama-7B`, `Llama2-7B`, `Falcon-7B` and `WizardLM-7B` models on conventional (`std`), `LoRA`, and `QLoRA` fine-tuning. The memory usage approximation is shown in Figure 6 (a). In conventional fine-tuning (`std`), the majority of memory is used to store the optimizer states (dark red) and gradients (orange). This results in massive memory consumption, exceeding the memory capacity of single `L40`, `A100`, and `H100`. On the other hand, single `MI300X` GPU provides plenty of memory to run conventional fine-tuning on these models. The paged Adam optimizer (Section 2.5.3) may be helpful in the situation to allow the fine-tuning to run even though the optimizer states cannot fit inside GPU memory at the expense of performance degradation due to excessive data movement between CPU and GPU.

Moving to `LoRA` fine-tuning, only the adapters are trainable, which significantly reduces the number of optimizer states and gradients. With `LoRA`, single `L40` is sufficient to fine-tune these models. This also opens the possibility to use high-end consumer-grade GPUs with memory in the range of 8 GB to 24 GB, making the models more accessible to the general public. Additional memory savings come from `QLoRA` by quantizing the pre-trained weights to a 4-bit floating-point format. However, since `LoRA` can already fit these models into single GPU, `QLoRA` may not be beneficial at this point. Its advantage becomes more apparent when fine-tuning large-size models.

4.1.2 Memory for Large-Size Models. Figure 5 (b) shows the trainable (purple) and frozen (grey) parameters for large models consisting of `Llama-65B`, `Llama2-70B`, `Falcon-40B` and `WizardLM-70B` on conventional (`std`), `LoRA`, and `QLoRA` fine-tuning. The memory usage approximation is shown in Figure 6 (b).

Like the smaller-size models, most of the memory stores the optimizer states and gradients. At this size, single GPU cannot sufficiently handle the memory demand for conventional fine-tuning (`std`). Thus, we look at the multi-GPU configuration on the compute platform. The R760xa with four `L40` GPUs cannot provide sufficient aggregate GPU memory to fine-tune these models conventionally. The XE9680 equipped with eight `H100` or eight `A100` GPUs may still be able to do conventional fine-tuning using the paged optimizer.

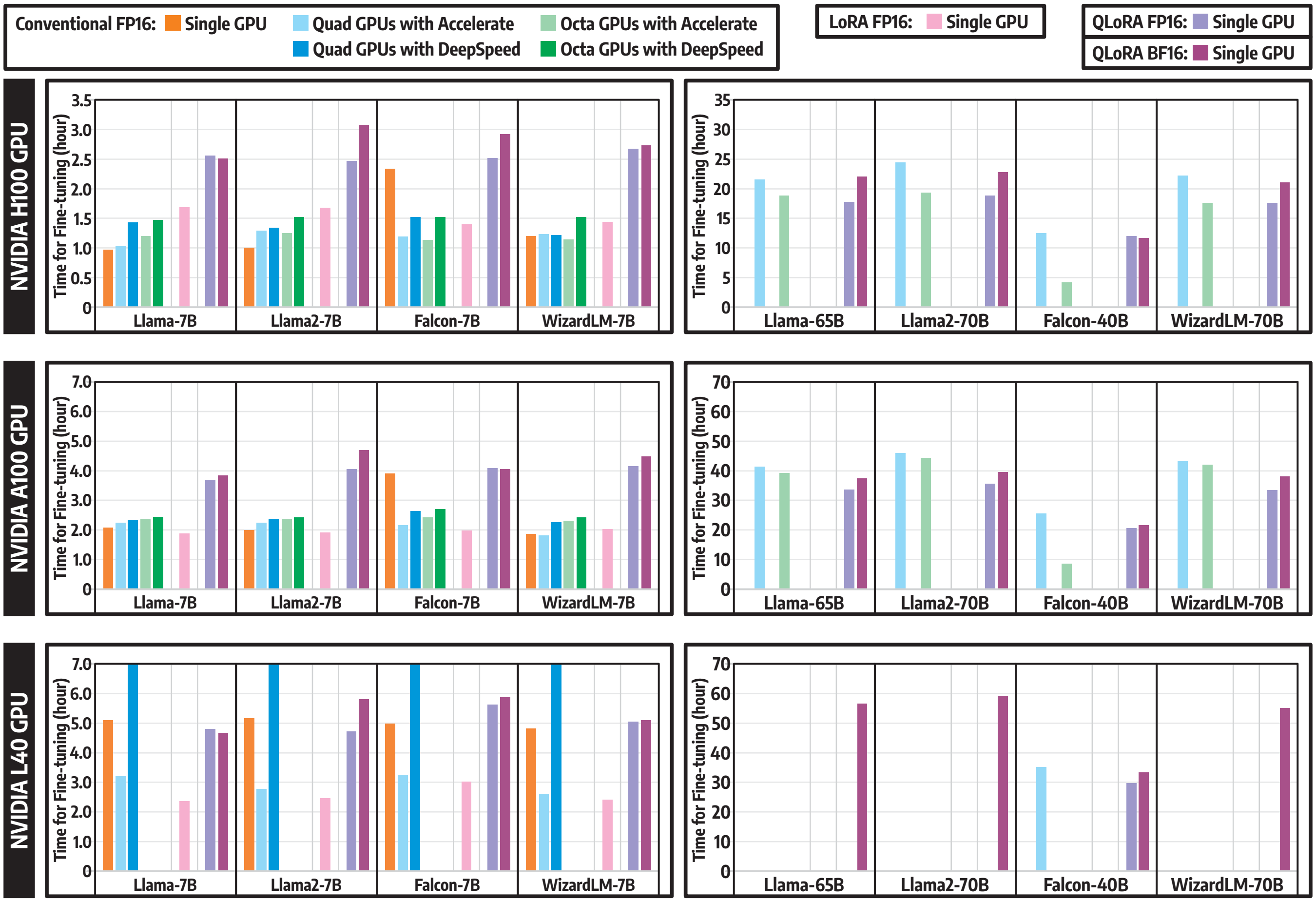

Figure 7: Time needed to fine-tune models with conventional (`std`), `LoRA`, and `QLoRA`. For conventional fine-tuning of large-size models, a multi-GPU setup (four or eight GPUs) is used with either HuggingFace Accelerate or Microsoft DeepSpeed Stage 2 backend. Note that when handling large-model sizes, only Accelerate and `QLoRA` can run on NVIDIA H100 and NVIDIA A100 GPUs and only `QLoRA` with BF16 can run on NVIDIA L40 GPU.

The XE9680, equipped with eight `MI300X` GPUs, provides sufficient aggregate memory to fine-tune these models.

While LoRA could fine-tune small-size models using a single GPU, this is no longer true with large-size models. Although memory requirements are significantly reduced, it still exceeds the memory capacity of single `L40`, `A100`, and `H100` GPU. Only `MI300X` GPU provides plenty of memory for `LoRA` fine-tuning for these models. Finally, `QLoRA` shines over `LoRA`, enabling fine-tuning large-size models with only one GPU. Both single `A100` and `H100` GPU can fine-tune these models using `QLoRA`. However, `L40` may not be able to fine-tune these models using `QLoRA` since its memory capacity is roughly the same as the estimated required memory, and additional memory is needed to store forward activations, dataset, and quantization constants excluded from the estimation.

4.2 Performance and Resource Utilization

In the previous section, we analyze and approximate the memory requirement of `LoRA` and `QLoRA` compared to conventional fine-tuning (`std`), giving us an idea of how significant the memory reduction offered by them. This section confirms the previous estimation by running the actual fine-tuning on four different GPUs for small-size models (i.e., 7 billion parameters) and large-size models (i.e., 40 billion parameters and above) to compare conventional (`std`), `LoRA`, and `QLoRA` fine-tuning in terms of performance (i.e., the time needed) and resource utilization (i.e., CPU and GPU memory). Figure 7 shows the time needed to fine-tune the models using different fine-tuning methods (i.e., `std`, `LoRA`, and `QLoRA`) on various GPUs (i.e., `H100`, `A100`, and `L40`).

4.2.1 Fine-tuning Performance for Small-Size Models. The left part of Figure 7 shows the time needed to fine-tune small-size models (i.e., `Llama-7B`, `Llama2-7B`, `Falcon-7B`, `WizardLM-7B`). Thanks to the Paged Adam Optimizer (Section 2.5.3), all GPUs can complete the conventional fine-tuning (`std`) process on small-size models. We previously estimated that it requires more memory than what is available in `H100`, `A100`, and `L40` GPUs (Section 4.1.1). The Paged Adam Optimizer leverages NVIDIA Unified Virtual Memory (UVM) to move the optimizer states between CPU and GPU memory. However, excessive data movement between CPU and GPU can degrade

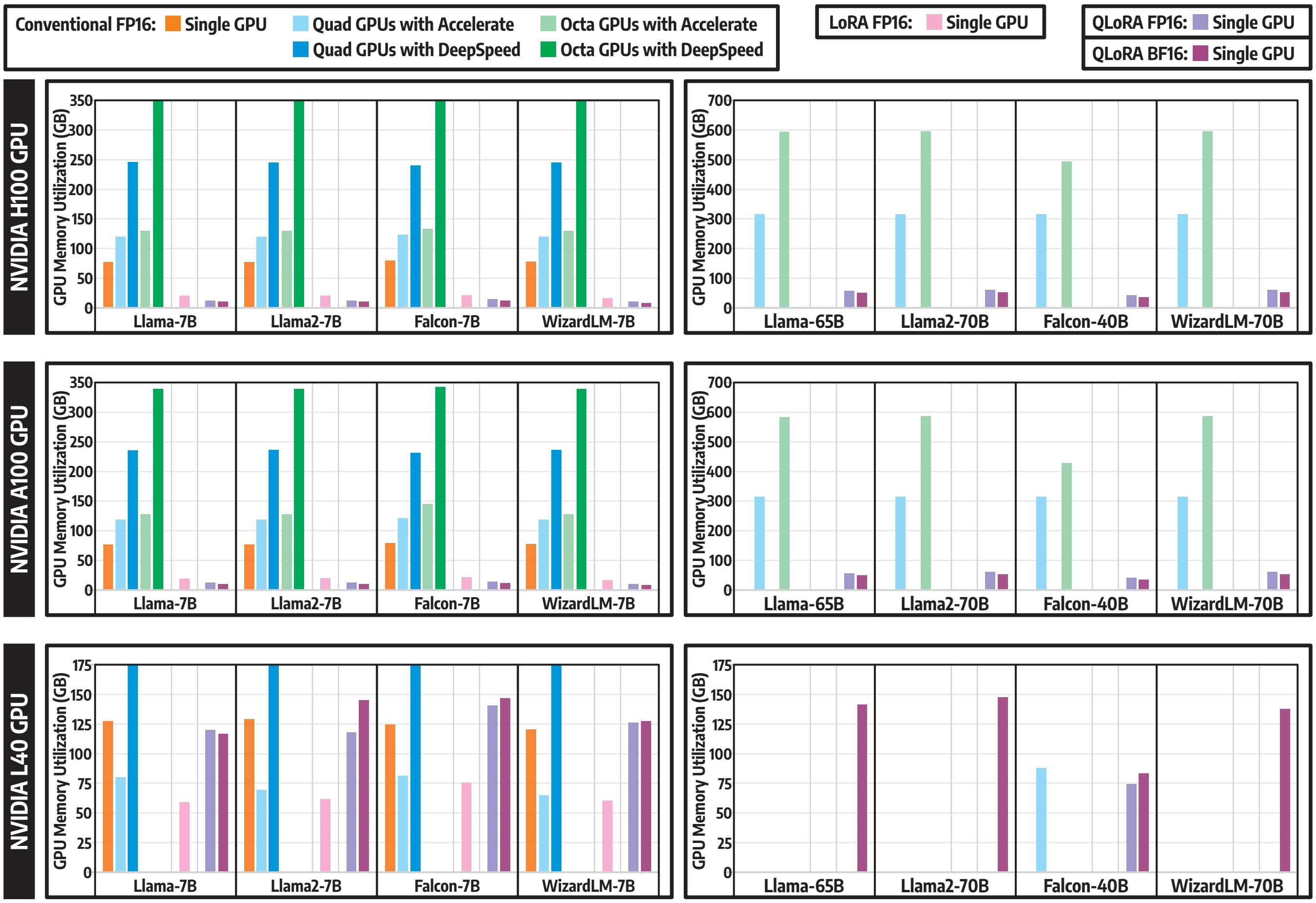

Figure 8: Aggregate GPU memory utilization during conventional (std), LoRA, and QLoRA fine-tuning. Note that when handling large-model sizes, only Accelerate and QLoRA can run on NVIDIA H100 and NVIDIA A100 GPUs and only QLoRA with BF16 can run on NVIDIA L40 GPU.

fine-tuning performance. The most notable impact of data movement is observed with L40 with only 48 GB available GPU memory.

Although single GPU is already sufficient for conventionally fine-tuning small-size models, we explore the performance improvements when using multiple GPUs with two different backends: HuggingFace Accelerate [27] and DeepSpeed ZeRO Stage 2 [75, 76]. With HuggingFace Accelerate to utilize multiple GPUs results in significant performance improvements for L40 since the model, optimizer states, and gradients are distributed among the available GPUs, reducing the need to transfer the data back and forth between CPU memory and GPU memory. However, for H100 and A100, adding more GPUs results in a slight performance degradation due to the communication overhead between GPUs. Single GPU already have sufficient memory to handle conventional fine-tuning.

On the other hand, the DeepSpeed ZeRO-2 backend partitions the optimizer states and gradient to remove data redundancy across GPUs, improving data parallelism. In this experiment, the offload features on DeepSpeed ZeRO are intentionally disabled. Compared to HuggingFace Accelerate, DeepSpeed ZeRO-2 performed slightly worse in H100 and A100, and performed significantly worse in L40. There are two reasons why DeepSpeed ZeRO-2 performed significantly worse in L40: 1) Inter-GPU communication in L40 must use the slower PCIe bus since it does not have a dedicated inter-GPU link (i.e., NVLink); and 2) DeepSpeed ZeRO-2 replicated the model to each GPU instead of partitioned it across GPU, causing the optimizer states to spill over to CPU memory due to smaller GPU memory in L40, which putting strain on the PCIe bus.

Moving into LoRA, the fine-tuning performance is improved for A100 and L40. Significant improvements are observed in L40 since there is no need to store parts of optimizer states in CPU memory, eliminating the data movement overhead. Finally, QLoRA performance is worse than conventional fine-tuning (std) and LoRA due to quantization and dequantization overhead. QLoRA with mixed precision FP32/FP16 (FP32 adapter) is observed to be faster than BF16 (BF16 adapter). The reason will be discussed in Section 4.3.2.

4.2.2 Fine-tuning Performance for Large-Size Models. The right part of Figure 7 shows the time needed to fine-tune large-size models (i.e., Llama-65B, Llama2-70B, Falcon-40B, WizardLM-70B). For a model this size, single GPU is no longer sufficient to fine-tune conventionally; instead, multiple GPUs are needed by utilizing either HuggingFace Accelerate or DeepSpeed ZeRO. DeepSpeed ZeRO-2 could not handle large-size models since it replicates the model parameters to each GPU instead of partitioning them across GPUs, leaving us with HuggingFace Accelerate. For H100 and A100, using eight GPUs yields slightly faster fine-tuning performance. Only Falcon-40B can be fine-tuned using four GPUs on L40. Next, LoRA is no longer sufficient for fine-tuning models this size using single GPU. Things become more interesting when models become larger, that is when QLoRA shines. QLoRA allows single GPU to fine-tune large-size models. Special mention goes into L40 where only QLoRA with BF16 can fit Llama-65B, Llama2-70B, and WizardLM-70B.

4.2.3 Memory Utilization. Figure 8 shows the aggregate GPU memory utilization for fine-tuning various models using various methods on various GPUs. In small-size models, LoRA gives an average GPU memory utilization of around 20 GB across all four models, which is a significant reduction from conventional fine-tuning. The memory utilization is further reduced by QLoRA with an average usage of 11.69 GB across all four models and GPU configurations. Although single GPU can no longer handle large-size models, QLoRA allows fine-tuning them with an average memory usage of 50 GB.

One may wonder about the advantage of choosing BF16 over FP16 in QLoRA, which gives slightly worse performance as discussed in Sections 4.2.1 and 4.2.2. QLoRA with BF16 (BF16 adapter) has an average GPU memory utilization of around 10.3 GB to fine-tune all small-size models. In contrast, QLoRA with FP16 (FP32 adapter) has an average GPU memory utilization of around 12.7 GB. This means that QLoRA with BF16 has almost 20% less memory demand than QLoRA with FP16, which will come into handy for larger model sizes. This is why only QLoRA with BF16 works on L40 when fine-tuning large-size models. Finally, CPU memory is highly utilized when Paged Optimizer is being used. For example, when running conventional fine-tuning on small-size models using L40, the CPU memory utilization reached 57 GB compared to 7 GB in H100 and A100. The highest CPU memory utilization is achieved when running conventional fine-tuning using HuggingFace Accelerate on large-size models: 526 GB and 499 GB utilization for four GPUs and eight GPUs run.

4.3 Model Quality Evaluation

While LoRA and QLoRA give promising advantages over conventional fine-tuning of LLMs in terms of memory requirements, infrastructure costs, and energy consumption, there is one puzzle left to complete the experiments: whether the fine-tuned model obtained using LoRA and QLoRA can compete with the conventional fine-tuning (std). To finish the puzzle, we use the Massive Multitask Language Understanding (MMLU) [34] benchmark to compare the quality of fine-tuned models on 57 subjects across STEM, social sciences, humanities, and more. Due to limited space, we only show the result of the MMLU benchmark for Llama2-7B and WizardLM-7B on Table 3 while other models follow the same pattern. Reviewing each subject on the MMLU benchmark would take too long, so we use the average MMLU accuracy score instead.

We also perform exhaustive experiments to determine the impact of data formats on model quality, fine-tuning runtime on A100, and GPU memory consumption. Regarding computation data format on std, LoRA, and QLoRA, we investigate both mixed-precision on FP16 or BF16. Specifically for QLoRA, we investigate the impact of 4-bit floating-point quantization using either FP4 or NF4. In addition, we also investigate the effect of using single quantization (SQ) and double quantization (DQ) as discussed in Section 2.5.2.

4.3.1 Average Accuracy Comparison. For Llama2-7B, the achieved accuracy score for conventional fine-tuning (std) is 0.24 for both FP16 and BF16. Both LoRA and QLoRA achieved an accuracy of 0.42 to 0.49, which is double what conventional fine-tuning can achieve. Without freezing the pre-trained weights, conventional fine-tuning on Llama2-7B may cause the model to lose some of its generalization and regularization, causing a drop in model accuracy after fine-tuning. On the other hand, conventional fine-tuning (std), LoRA, and QLoRA achieved on-par accuracy in WizardLM-7B.

4.3.2 Mixed-Precision using FP16 or BF16. The BF16 data format is supposed to give better performance than mixed-precision fine-tuning using FP32/FP16 for reasons: 1) BF16 already provides the same dynamic range as FP32, and thus there is no need to use larger data formats, reducing the memory and bandwidth demands and 2) matrix units (i.e., Tensor Cores on NVIDIA GPUs) inside the GPUs support BF16, which should give tremendous speed-up over FP32. However, we observe the opposite way: the mixed-precision fine-tuning using FP32/FP16 is slightly faster than BF16 for conventional (std), LoRA, and QLoRA fine-tuning. One reason is the automatic demotion of FP32 to TensorFloat32 (TF32) [15] by the CUDA libraries under the hood to take advantage of the matrix units, giving comparable performance to BF16 without the overhead of type-casting. Regarding memory usage, using BF16 does not lower the memory usage on conventional fine-tuning. On the other hand, on LoRA and QLoRA, we see lower memory usage when using BF16 by as much as 5% and 15%, respectively. This is the reason why single L40 can run QLoRA fine-tuning for Llama-65B, Llama2-70B, and WizardLM-70B models (Section 4.2).

4.3.3 FP4 and NF4 Comparison on QLoRA. In terms of quality, the NF4 achieves a slightly better average score than FP4: less than 4% average score for Llama2-7B and on-par on WizardLM-7B. However, NF4 has more computation overhead due to the quantization based on empirical distribution, resulting in roughly 4%-5% longer time for fine-tuning the model. There are no differences in memory usage between these formats.

4.3.4 Single and Double Quantization Comparison on QLoRA. In terms of quality, both single (SQ) and double (DQ) give roughly the same score on the MMLU benchmark. Double quantization saves memory usage by up to 5%, which is very useful when handling larger models. However, double quantization involves more computation overhead, resulting in around 2%-4% longer time.

4.4 Preliminary Evaluation on MI300X

This section discusses the preliminary evaluation of MI300X, where, in this case, the libraries are ported using the provided compiler, and no optimization is given (Section 3.1). The MI300X is anticipated to be more popular for handling large language models, so having a look at the early availability of software libraries is important.

4.4.1 Fine-tuning Small-Size Models. As shown in Figure 9 (a), DeepSpeed ZeRO-2 performed better than HuggingFace Accelerate in MI300X. Porting HuggingFace Accelerate [27] using the compiler

Table 3: MMLU Accuracy Score for `Llama2-7B` and `WizardLM-7B`

Measurements	`Llama2-7B`											
	`Std`		`LoRA`		`QLoRA`							
					NF4 Quantization				FP4 Quantization			
					FP16		BF16		FP16		BF16	
	FP16	BF16	FP16	BF16	SQ	DQ	SQ	DQ	SQ	DQ	SQ	DQ
Fine-tuning Time (H:MM)	2:12	2:15	2:37	2:39	5:19	5:25	5:35	5:39	5:03	5:11	5:17	5:22
GPU Memory (GB)	78.1	78.7	19.4	18.5	14.6	14.3	12.5	12.3	14.6	14.3	12.5	12.3
Average MMLU Accuracy Scores	0.24	0.24	0.49	0.47	0.47	0.46	0.46	0.45	0.42	0.44	0.44	0.46

Measurements	`WizardLM-7B`											
	`Std`		`LoRA`		`QLoRA`							
					NF4 Quantization				FP4 Quantization			
					FP16		BF16		FP16		BF16	
	FP16	BF16	FP16	BF16	SQ	DQ	SQ	DQ	SQ	DQ	SQ	DQ
Fine-tuning Time (H:MM)	2:11	2:14	2:31	2:34	5:02	5:08	5:16	5:21	4:48	4:54	4:58	5:03
GPU Memory (GB)	78	78.6	16.7	15.8	8.9	8.5	7.3	7.1	8.9	8.4	7.3	7.1
Average MMLU Accuracy Scores	0.26	0.26	0.26	0.27	0.26	0.26	0.27	0.27	0.26	0.26	0.27	0.27

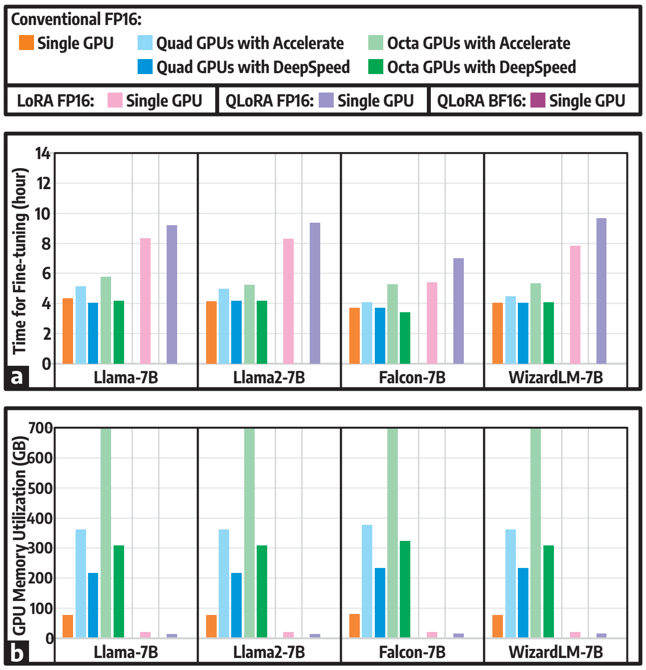

Figure 9: Preliminary fine-tuning performance (a) and GPU memory utilization (b) of `MI300X`.

may not yield optimized code to run on `MI300X`. Next, `QLoRA` with BF16 cannot run since the BitsandBytes [17] could not detect BF16 hardware support even though `MI300X` has native support for it.

4.4.2 Fine-tuning Large-Size Models. The chart for large-size models is not shown since only `QLoRA` with FP16 can run for fine-tuning large-size models. The `QLoRA` with BF16 cannot run for the same reason as Small-Size models. The HuggingFace Accelerate fails to run on `MI300X` when fine-tuning large-size models due to undesired behavior. Instead of splitting and distributing model parameters across the GPUs, it replicates them. The replication results in significantly higher aggregate memory utilization. This is why Accelerate faces out-of-memory errors when handling large-size models.

This undesired behavior with Accelerate can also be observed in small-size models, as shown in Figure 9 (b). Compared to single GPU (orange), the quad GPUs with Accelerate (light blue) and the octal GPUs with Accelerate (light green) consume four and eight times the memory. The desired behavior of Accelerate is shown in Figure 8, where the quad GPUs (light blue) and octal GPUs (light green) aggregate memory utilization are not significantly different. The small differences are due to the additional memory required for communication buffers between GPUs.

4.4.3 Discussion on Optimized Libraries. The `MI300X` provides immense computing power and memory, as shown in Table 1. Since it is still new to the market, many libraries may have not been fully optimized for newer hardware. Relying on the compiler alone to port the codes is not sufficient. It takes time for the libraries to catch up with new hardware architecture and become more mature which is essential to unleash the performance potential. As more people have access to `MI300X`, many developers will have their libraries optimized for it, such as the VLLM project [49], SGLang [106], and Lamini [51]. Not only does it help to get the performance out of the hardware, but it also eliminates the undesired behavior of the libraries when they are used in newer computing hardware.

5 Conclusion

With the increasing size of state-of-the-art models, the cost of fine-tuning the models skyrockets as they require large computing infrastructures and high energy usage. It is necessary to find solutions to efficiently fine-tune large models, allowing lower costs of adopting such models for organizations and reducing the impact on the environment. LoRA is one of the proposed solutions to efficiently fine-tune large models by leveraging the fact that the model has a low intrinsic dimension during fine-tuning. Even with LoRA, the required memory may still be out of reach for resource-constrained computing infrastructure. QLoRA improves LoRA with three primary innovations: four-bit quantization of the pre-trained models, double quantization, and paged optimizers. These three innovations allow QLoRA to reduce memory requirements by as much as 75% compared to LoRA, allowing single GPU with 48 GB and 80 GB of memory to fine-tune 70 billion parameter models. In addition, QLoRA delivers model quality that is on par with or exceeds the quality of the model obtained from standard fine-tuning. Although QLoRA is still in its infancy at the time of writing, it will see significant adoption from the community. The integration with the popular HuggingFace software stack facilitates the easy use of QLoRA to existing fine-tuning flow. Finally, while AMD MI300X GPUs provide the largest memory capacity and compute power, a lot of efforts need to be made to optimize software and libraries to get the most out of the hardware.

Acknowledgments

This research was supported in part by the United States National Science Foundation Grants #2326894, #2425655, the Semiconductor Research Corporation (SRC) Task 3148.001, Texas Advanced Computing Center (TACC) and NVIDIA Applied Research Accelerator Program Grant. Any opinions, findings, conclusions, or recommendations are those of the authors and not the funding agencies. The authors would like to express their sincere gratitude to Stephen Rousset, Liz Raymond, and Daniel Amann of Dell Technologies for their support and assistance during this work. The authors would also like to thank the anonymous reviewers for their constructive feedback and suggestions.

References

[1] Advanced Micro Devices. 2023. *AMD CDNA™ 2 Architecture: The All-New AMD GPU Architecture for the Modern Era of HPC and AI.* Whitepaper. Advanced Micro Devices, California, US. https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

[2] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 7319–7328. https://doi.org/10.18653/v1/2021.acl-long.568

[3] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The Falcon Series of Open Language Models. arXiv:2311.16867 [cs.CL]

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 173–182. https://proceedings.mlr.press/v48/amodei16.html

[5] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. arXiv:2007.03051 [cs.CY]

[6] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. 2023. The Power of Generative AI: A Review of Requirements, Models, Input-Output Formats, Evaluation Metrics, and Challenges. *Future Internet* 15, 8 (2023), 60 pages. https://doi.org/10.3390/fi15080260

[7] Michele Banko and Eric Brill. 2001. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (Toulouse, France) *(ACL '01)*. Association for Computational Linguistics, USA, 26–33. https://doi.org/10.3115/1073012.1073017

[8] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. arXiv:1903.10676 [cs.CL]

[9] Lenz Belzner, Thomas Gabor, and Martin Wirsing. 2024. Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study. In *Bridging the Gap Between AI and Reality*, Bernhard Steffen (Ed.). Springer Nature Switzerland, Cham, 355–374.

[10] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*. International Conference on Learning Representations, Appleton, WI, USA, 35 pages. https://openreview.net/forum?id=B1xsqj09Fm

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

[12] Alexander E.I Brownlee, Jason Adair, Saemundur O. Haraldsson, and John Jabbo. 2021. Exploring the Accuracy – Energy Trade-off in Machine Learning. In *2021 IEEE/ACM International Workshop on Genetic Improvement (GI)*. IEEE, Madrid, Spain, 11–18. https://doi.org/10.1109/GI52543.2021.00011

[13] Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. 2023. Broken Neural Scaling Laws. In *The Eleventh International Conference on Learning Representations*. International Conference on Learning Representations, Appleton, WI, USA, 32 pages. https://openreview.net/forum?id=sckjveqlCZ

[14] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun. 2023. A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT. arXiv:2303.04226 [cs.AI]

[15] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. 2021. NVIDIA A100 Tensor Core GPU: Performance and Innovation. *IEEE Micro* 41, 2 (2021), 29–35. https://doi.org/10.1109/MM.2021.3061394

[16] KATE CRAWFORD. 2021. *The Atlas of AI.* Yale University Press, New Haven, Connecticut, U.S. https://doi.org/10.2307/j.ctv1ghv45t

[17] Tim Dettmers. 2023. bitsandbytes. https://github.com/TimDettmers/bitsandbytes

[18] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG]

[19] Christof Ebert and Panos Louridas. 2023. Generative AI for Software Practitioners. *IEEE Software* 40, 4 (2023), 30–38. https://doi.org/10.1109/MS.2023.3265877

[20] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, and Pierre Zweigenbaum. 2022. Re-train or train from scratch? Comparing pre-training strategies of BERT in the medical domain. In *LREC 2022 - Language Resources and Evaluation Conference (Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022))*. European Language Resources Association (ELRA), Marseille, France, 2626–2633. https://hal.science/hal-03803880

[21] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. arXiv:2310.03533 [cs.SE]

[22] Lizhou Fan, Lingyao Li, Zihui Ma, Sanggyu Lee, Huizi Yu, and Libby Hemphill. 2023. A Bibliometric Review of Large Language Models Research from 2017 to 2023. arXiv:2304.02020 [cs.DL]

[23] Emilio Ferrara. 2023. GenAI Against Humanity: Nefarious Applications of Generative Artificial Intelligence and Large Language Models. arXiv:2310.00737 [cs.CY]

[24] Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernian, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Christopher Olah, Dario Amodei, and Jack Clark. 2022. Predictability and Surprise in Large Generative Models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency* (Seoul, Republic of Korea) *(FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 1747–1764. https://doi.org/10.1145/3531146.3533229

[25] Henry Gouk, Timothy Hospedales, and massimiliano pontil. 2021. Distance-Based Regularisation of Deep Networks for Fine-Tuning. In *International Conference on Learning Representations*. International Conference on Learning Representations, Appleton, WI, USA, 11 pages. https://openreview.net/forum?id=IFqrg1p5Bc

[26] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchan. 2023. ChatGPT is not all you need. A State of the Art Review of large Generative AI models. arXiv:2301.04655 [cs.LG]

[27] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate

[28] Neeraj Gupta. 2021. A Pre-Trained Vs Fine-Tuning Methodology in Transfer Learning. *Journal of Physics: Conference Series* 1947, 1 (jun 2021), 012028. https://doi.org/10.1088/1742-6596/1947/1/012028

[29] Muhammad Usman Hadi, qasem al tashi, Rizwan Qureshi, Abbas Shah, amgad muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, and Seyedali Mirjalili. 2023. Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects. *TechRxiv e-Prints* 1 (Nov. 2023), 44 pages. https://doi.org/10.36227/techrxiv.23589741.v4

[30] Bagus Hanindhito. 2025. *hibagus/QLoRA-Experiment: Camera-Ready Version.* The University of Texas at Austin and Dell Technologies. https://doi.org/10.5281/zenodo.14962603

[31] Bagus Hanindhito and Lizy K. John. 2024. Accelerating ML Workloads using GPU Tensor Cores: The Good, the Bad, and the Ugly. In *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering* (London, United Kingdom) *(ICPE '24)*. Association for Computing Machinery, New York, NY, USA, 12 pages. https://doi.org/10.1145/3629526.3653835

[32] William Grant Hatcher and Wei Yu. 2018. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* 6 (2018), 24411–24432. https://doi.org/10.1109/ACCESS.2018.2830661

[33] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2023. A Survey of Large Language Models for Healthcare: from Data, Technology, and Applications to Accountability and Ethics. arXiv:2310.05694 [cs.CL]

[34] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. arXiv:2009.03300 [cs.CY]

[35] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. 2020. Scaling

Laws for Autoregressive Generative Modeling. arXiv:2010.14701 http://arxiv.org/abs/2010.14701
- [36] Joel Hestness, Newsha Ardalani, and Gregory Diamos. 2019. Beyond Human-Level Accuracy: Computational Challenges in Deep Learning. In *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming* (Washington, District of Columbia) *(PPoPP '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3293883.3295710
- [37] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep Learning Scaling is Predictable, Empirically. arXiv:1712.00409 [cs.LG]
- [38] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (nov 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
- [39] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL]
- [40] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]
- [41] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. arXiv:2212.10403 [cs.CL]
- [42] Mikhail Isaev, Nic McDonald, and Richard Vuduc. 2023. Scaling Infrastructure to Support Multi-Trillion Parameter LLM Training. In *Architecture and System Support for Transformer Models (ASSYST @ISCA 2023)*. Architecture and System Support for Transformer Models (ASSYST) Workshop, Orlando, FL, USA, 5 pages. https://openreview.net/forum?id=rqn2v1Ltgn0
- [43] Won Jeon, Gun Ko, Jiwon Lee, Hyunwuk Lee, Dongho Ha, and Won Woo Ro. 2021. Chapter Six - Deep learning with GPUs. In *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, Shiho Kim and Ganesh Chandra Deka (Eds.). Advances in Computers, Vol. 122. Elsevier, Amsterdam, Netherlands, 167–215. https://doi.org/10.1016/bs.adcom.2020.11.003
- [44] Kun Jing and Jungang Xu. 2019. A Survey on Neural Network Language Models. arXiv:1906.03591 [cs.CL]
- [45] Mlađan Jovanović and Mark Campbell. 2022. Generative Artificial Intelligence: Trends and Prospects. *Computer* 55, 10 (2022), 107–112. https://doi.org/10.1109/MC.2022.3192720
- [46] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG]
- [47] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]
- [48] Will Knight. 2023. OpenAI's CEO Says the Age of Giant AI Models Is Already Over. https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/
- [49] 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* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165
- [50] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. OpenAssistant Conversations – Democratizing Large Language Model Alignment. arXiv:2304.07327 [cs.CL]
- [51] Lamini Inc. 2023. Introducing Lamini, the LLM Platform for Rapidly Customizing Models. https://www.lamini.ai/blog/introducing-lamini
- [52] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (Feb. 2020), 1234–1240.
- [53] Chuan Li. 2020. OpenAI's GPT-3 Language Model: A Technical Overview. https://lambdalabs.com/blog/demystifying-gpt-3
- [54] Dongyue Li and Hongyang Zhang. 2021. Improved Regularization and Robustness for Fine-tuning in Neural Networks. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). Neural Information Processing Systems, San Diego, CA, USA, 14 pages. https://openreview.net/forum?id=j7buX9nsfis
- [55] Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Xiangyang Ji, Antoni Chan, and Rong Jin. 2022. Improved Fine-Tuning by Better Leveraging Pre-Training Data. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Neural Information Processing Systems, San Diego, CA, USA, 32568–32581. https://proceedings.neurips.cc/paper_files/paper/2022/file/d1c88f9790765146ec8fb5d02e5653a0-Paper-Conference.pdf
- [56] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft
- [57] Sparsh Mittal and Shraiysh Vaishay. 2019. A survey of techniques for optimizing deep learning on GPUs. *Journal of Systems Architecture* 99 (2019), 101635. https://doi.org/10.1016/j.sysarc.2019.101635
- [58] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A Comprehensive Overview of Large Language Models. arXiv:2307.06435 [cs.CL]
- [59] Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra, and Guy Boudoukh. 2017. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '17)*. Association for Computing Machinery, New York, NY, USA, 5–14. https://doi.org/10.1145/3020078.3021740
- [60] NVIDIA Corporation. 2017. *NVIDIA Tesla V100 GPU Architecture: The World's Most Advanced Data Center GPU.* Whitepaper. NVIDIA Corporation, California, US. https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
- [61] NVIDIA Corporation. 2020. *NVIDIA A100 Tensor Core GPU Architecture: Unprecedented Acceleration at Every Scale.* Whitepaper. NVIDIA Corporation, California, US. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf
- [62] NVIDIA Corporation. 2022. *NVIDIA H100 Tensor Core GPU Architecture: Exceptional Performance, Scalability, and Security for The Data Center.* Whitepaper. NVIDIA Corporation, California, US. https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper
- [63] NVIDIA Corporation. 2023. *NVIDIA L40: Delivering unprecedented visual computing performance for the data center.* Datasheet. NVIDIA Corporation, US.
- [64] NVIDIA Corporation. 2024. *NVIDIA DGX B200: A Unified AI Platform for Training, Fine-tuning, and Inference.* Datasheet. NVIDIA Corporation, California, US. https://resources.nvidia.com/en-us-dgx-systems/dgx-b200-datasheet
- [65] NVIDIA Corporation. 2024. *NVIDIA H200 Tensor Core GPU: Supercharging AI and HPC Workloads.* Datasheet. NVIDIA Corporation, California, US. https://nvdam.widen.net/s/nb5zzzsjdf/hpc-datasheet-sc23-h200-datasheet-3002446
- [66] Saurabh Pahune and Manoj Chandrasekharan. 2023. Several Categories of Large Language Models (LLMs): A Short Survey. *International Journal for Research in Applied Science and Engineering Technology* 11, 7 (July 2023), 615–633. https://doi.org/10.22214/ijraset.2023.54677
- [67] Nikolaos Pappas and Thomas Meyer. 2012. A Survey on Language Modeling using Neural Networks. *Infoscience: EPFL Scientific Publications* 1 (2012), 23 pages. http://infoscience.epfl.ch/record/192566
- [68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG]
- [69] Dylan Patel and Gerald Wong. 2023. GPT-4 Architecture, Infrastructure, Training Dataset, Costs, Vision, MoE. https://www.semianalysis.com/p/gpt-4-architecture-infrastructure?utm_source=%2Fsearch%2FGPT-4&utm_medium=reader2
- [70] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs.LG]
- [71] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202
- [72] Behnam Pourghassemi, Chenghao Zhang, Joo Hwan Lee, and Aparna Chandramowlishwaran. 2020. On the Limits of Parallelizing Convolutional Neural Networks on GPUs. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures* (Virtual Event, USA) *(SPAA '20)*. Association for Computing Machinery, New York, NY, USA, 567–569. https://doi.org/10.1145/3350755.3400266
- [73] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [74] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (jan 2020), 67 pages.

[75] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory Optimizations toward Training Trillion Parameter Models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '20)*. IEEE Press, Atlanta, Georgia, Article 20, 16 pages.

[76] 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* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3505–3506. https://doi.org/10.1145/3394486.3406703

[77] Partha Pratim Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* 3 (2023), 121–154. https://doi.org/10.1016/j.iotcps.2023.04.003

[78] Saqib Ul Sabha, Assif Assad, Nusrat Mohi Ud Din, and Muzafar Rasool Bhat. 2024. From scratch or pretrained? An in-depth analysis of deep learning approaches with limited data. *International Journal of System Assurance Engineering and Management* 1 (29 Apr 2024), 0 pages. https://doi.org/10.1007/s13198-024-02345-4

[79] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. 2022. Compute Trends Across Three Eras of Machine Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Padua, Italy, 10 pages. https://doi.org/10.1109/ijcnn55064.2022.9891914

[80] Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The Cost of Training NLP Models: A Concise Overview. arXiv:2004.08900 [cs.CL]

[81] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

[82] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. arXiv:2201.11990 [cs.CL]

[83] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. arXiv:1906.02243 [cs.CL]

[84] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A Large Language Model for Science. arXiv:2211.09085 [cs.CL]

[85] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature Medicine* 29, 8 (01 Aug 2023), 1930–1940. https://doi.org/10.1038/s41591-023-02448-8

[86] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[87] 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]

[88] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. https://doi.org/10.1145/3491101.3519665

[89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Neural Information Processing Systems, San Diego, CA, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[90] 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

[91] Meng Wang, Weijie Fu, Xiangnan He, Shijie Hao, and Xindong Wu. 2022. A Survey on Large-Scale Machine Learning. *IEEE Transactions on Knowledge and Data Engineering* 34, 6 (2022), 2574–2594. https://doi.org/10.1109/TKDE.2020.3015777

[92] Shibo Wang and Pankaj Kanwar. 2019. BFloat16: The secret to high performance on Cloud TPUs. *Google Cloud Blog* 4 (2019), 1 pages.

[93] 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 M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[94] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S. Yu. 2023. Multimodal Large Language Models: A Survey. arXiv:2311.13165 [cs.AI]

[95] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. arXiv:2303.17564 [cs.LG]

[96] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of Automatica Sinica* 10, 5 (2023), 1122–1136. https://doi.org/10.1109/JAS.2023.123618

[97] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv:2309.07864 [cs.AI]

[98] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. arXiv:2304.12244 [cs.CL]

[99] Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E. Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Anthony B. Costa, Mona G. Flores, Ying Zhang, Tanja Magoc, Christopher A. Harle, Gloria Lipori, Duane A. Mitchell, William R. Hogan, Elizabeth A. Shenkman, Jiang Bian, and Yonghui Wu. 2022. A large language model for electronic health records. *npj Digital Medicine* 5, 1 (26 Dec 2022), 194. https://doi.org/10.1038/s41746-022-00742-2

[100] Yi Yang, Mark Christopher Siy UY, and Allen Huang. 2020. FinBERT: A Pretrained Language Model for Financial Communications. arXiv:2006.08097 [cs.CL]

[101] Yaodong Yu, Heinrich Jiang, Dara Bahri, Hossein Mobahi, Seungyeon Kim, Ankit Singh Rawat, Andreas Veit, and Yi Ma. 2022. An Empirical Study of Pre-trained Models on Out-of-distribution Generalization. https://openreview.net/forum?id=2RYOwBOFesi

[102] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. 2023. Large Language Models for Robotics: A Survey. arXiv:2311.07226 [cs.RO]

[103] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. 2023. Large language models for human–robot interaction: A review. *Biomimetic Intelligence and Robotics* 3, 4 (2023), 100131. https://doi.org/10.1016/j.birob.2023.100131

[104] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. 2023. Planning with Large Language Models for Code Generation. arXiv:2303.05510 [cs.LG]

[105] 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:2303.18223 [cs.CL]

[106] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient Execution of Structured Language Model Programs. arXiv:2312.07104 [cs.AI] https://arxiv.org/abs/2312.07104

[107] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large Language Models for Information Retrieval: A Survey. arXiv:2308.07107 [cs.CL]