The nextAI Solution to the NeurIPS LLM Efficiency Challenge

Park, Gyu-Won **Shin, Dong-Il** **Oh, Sol-Gil** **Ryu, Sang-Gi** **Kim, Byung-Hak**

Abstract

The rapid evolution of Large Language Models (LLMs) has significantly impacted the field of natural language processing, but their growing complexity raises concerns about resource usage and transparency. Addressing these challenges, we participated in the NeurIPS LLM Efficiency Challenge, aiming to fine-tune a foundation model within stringent constraints. Our focus was the LLaMa2 70 billion model, optimized on a single A100 40GB GPU within a 24-hour limit. Our methodology hinged on a custom dataset, carefully assembled from diverse open-source resources and benchmark tests, aligned with the challenge's open-source ethos. Our approach leveraged Quantized-Low Rank Adaptation (QLoRA) Fine tuning, integrated with advanced attention mechanisms like Flash Attention 2. We experimented with various configurations of the LoRA technique, optimizing the balance between computational efficiency and model accuracy. Our fine-tuning strategy was underpinned by the creation and iterative testing of multiple dataset compositions, leading to the selection of a version that demonstrated robust performance across diverse tasks and benchmarks. The culmination of our efforts was an efficiently fine-tuned LLaMa2 70B model that operated within the constraints of a single GPU, showcasing not only a significant reduction in resource utilization but also high accuracy across a range of QA benchmarks. Our study serves as a testament to the feasibility of optimizing large-scale models in resource-constrained environments, emphasizing the potential of LLMs in real-world applications.

1 Introduction

The emergence and rapid advancement of LLMs (OpenAI [32], Touvron et al. [42]) have revolutionized various domains of natural language processing. As these models have grown in complexity and capability, their parameter counts have surged exponentially. Such massive models (Chowdhery et al. [4], Brown et al. [1]), although impressive in their performance, come with their share of challenges. First, their ever-increasing size necessitates the use of extensive GPU resources, leading to escalated costs and reduced accessibility for the majority. More recently, there has been a concerning trend where the specifics of the model architectures and their training methodologies are not disclosed, resulting in a lack of transparency and reproducibility. There is a pressing need for clear and reproducible training procedures, which has been conspicuously absent in many recent developments.

In response to these concerns, the NeurIPS LLM Efficiency Challenge has been introduced. This challenge focuses on adapting a foundation model to specific tasks but with imposed constraints. Participants are required to fine-tune models on a single GPU, either of the 4090 or A100 (40GB) variants, all within a strict 24-hour limit. Such constraints emphasize real-world applicability, simulating scenarios of limited resources. Furthermore, the initiative highlights the significance of open source, promoting not just replicability but also widespread accessibility, thereby eliminating barriers of proprietary restrictions.

In addressing this challenge, our team at nextAI identified a model that thrived under the set constraints. While several models like LLaMA-30b, LLaMA-65b (Touvron et al. [41]) and Mistral-7b (Jiang et al. [16]) were considered, LLaMA2-13b, LLaMA2-70b stood out, and we fine-tuned it on a single A100 40GB GPU within 24 hours. Key to this achievement were techniques like Quantization, Parameter-Efficient Fine-Tuning (Dettmers et al. [10], Hu et al. [15], Mangrulkar et al. [27]) and advanced Attention mechanisms (Dao [7],Dao et al. [8], Liu et al. [25], Vaswani et al. [43]). Our custom dataset (Lee et al. [17], , Zhang et al. [48], Zhou et al. [49]) further optimized the fine-tuning process. This approach emphasized the adaptability of LLMs in constrained environments.

Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar et al. [27]) enables efficient adaptation of pre-trained language models to downstream applications by fine-tuning a subset of model parameters, reducing computational and storage challenges. Among the available methods, we primarily employed LoRA (Low-Rank Adaptation of Large Language Models) (Hu et al. [15]). Thanks to the advancements in this domain, we were able to leverage Quantized-LoRA (QLoRA) (Dettmers et al. [10]), which combined quantization with LoRA, enhancing our model's efficiency. Quantization techniques offer significant benefits by reducing the model size and accelerating inference, making it possible to deploy more complex models on resource-constrained devices with minimal loss of accuracy. Notably, certain QLoRA experiments, such as variations with changing lora_r values and differing LoRA layer targets, lacked exhaustive results. Our team bridged this gap, providing in-depth insights beneficial for subsequent research in the field.

Transformers have come a long way since "Attention is All You Need" (Vaswani et al. [43]) revolutionized the field. As models grow in complexity and the volume of attention computations surges, efficiently managing these computational demands becomes crucial. Our solution is the adoption of the Flash Attention framework (Dao [7], Dao et al. [8]), which streamlines attention calculations through an innovative IO-aware algorithm, optimizing the synergy between GPU high bandwidth memory (HBM) and on-chip SRAM. This advancement is pivotal for maintaining performance despite the increased complexity of attention mechanisms in larger models. While the Sliding Window Attention (SWA) (Liu et al. [25]) mechanism also contributes to computational efficiency, it's the incorporation of Flash Attention that stands out as a transformative measure to sustain model scalability and computational tractability.

Our training strategy hinged on the exclusive use of open-source data, assembling a corpus that aligns with our model's training needs. Influenced by Chung et al. [5], we crafted a dataset with a spectrum of instructions for robust context coverage. Insights from top performers on the HuggingFace Open LLM Leaderboard refined our dataset design. The composition details are in Table 1. We adopted the Alpaca format (Taori et al. [39]) for instruction tuning, leveraging its effectiveness in task comprehension and execution. This dual approach of dataset diversity and instructional clarity is the foundation for our ensuing performance analysis.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

In the evolving field of LLMs, a suite of PEFT techniques has significantly diminished the necessity for comprehensive model fine-tuning. Among these, LoRA (Hu et al. [15]) stands out by adapting models through low-rank matrix updates, while methods like Prefix Tuning (Li and Liang [19]) enhance model performance by optimizing continuous prompts. Meanwhile, P-Tuning (Liu et al. [24]) illustrates the untapped potential of LLMs to comprehend complex contexts with minimal parameter adjustments. Other methods, including Prompt Tuning (Lester et al. [18]) and IA3 (Liu et al. [23]), focus on refining prompts and modulating inner activations, respectively.

The publication "Towards A Unified View of Transfer Learning" (He et al. [13]) offers a substantial contribution to the field of transfer learning for LLMs. It introduces a comprehensive framework that reframes existing approaches as precise modifications to the hidden states of pretrained models. This framework outlines various design dimensions, such as the specific functions for computing modifications and their points of application, facilitating more sophisticated and effective transfer learning tactics. This integrated approach offers a nuanced perspective on tailoring PEFT methods to leverage the innate strengths of LLMs.

2.2 Quantization and LoRA

In tandem with the growth of LLMs, quantization has emerged as a pivotal compression technique, enhancing efficiency and scalability by reducing the bit width of parameters and activations (Dettmers et al. [9], Dettmers et al. [10]). Efforts have primarily centered on maintaining the fidelity of LLMs post-quantization, with a focus on minimizing the computational load without necessitating retraining. Innovative strides in the field of LLM quantization have been made to mitigate the impact of outliers in parameter distributions, fueling the development of both adaptive and dynamic quantization techniques, as well as sophisticated clustering methods that apply customized quantization to various parameter sets. Alongside these strides, the integration of LoRA with quantization techniques has been a game-changer, notably with QLoRA demonstrating marked efficiency gains. Building upon this, QA-LoRA (Xu et al. [46]) has accelerated processing speeds. Meanwhile, LongLoRA (Chen et al. [3]) has effectively addressed the efficiency issues encountered in fine-tuning with longer contexts.

2.3 Attention and Memory

Transformers (Vaswani et al. [43]) struggle with the computational intensity of self-attention for long sequences, but Flash Attention (Dao et al. [8]) mitigates this by optimizing memory exchanges within GPU architecture, significantly enhancing efficiency. The subsequent Flash Attention version 2 (Dao [7]) further refines this approach, with better work partitioning and parallelization strategies, halving the computational demands and nearing the efficiency of optimized matrix multiplications, a step eagerly adopted by the open-source community.

Advancing beyond Flash Attention's efficiencies, innovative methods like LLaMA2's Grouped-query attention (GQA) ([42]) and Mistral's (Jiang et al. [16]) Sliding Window Attention (SWA) (Liu et al. [25]) have emerged, offering faster inference and managing longer sequences with reduced computational load. Additionally, LongLoRA's Shift Short Attention (Chen et al. [3]) has optimized attention computation through intelligent structuring. Together, these advancements represent substantial strides toward minimizing GPU memory usage and shortening computational duration.

2.4 Instruction Tuning

Instruction tuning in contemporary research broadly divides into two primary strategies. The first leverages data integration from annotated natural language datasets, exemplified by the likes of the FLAN (Wei et al. [45]), where text-label pairs are ingeniously transformed into (instruction, output) pairs via templates. The second paradigm shifts towards generating datasets with the aid of LLMs such as GPT-3.5-Turbo or GPT-4 (OpenAI [32]). This method capitalizes on the efficiency of LLMs to produce outputs from both manually curated and LLM-expanded instructions, giving rise to datasets such as InstructionWild (Ni et al. [30]) and Self-Instruct (Wang et al. [44]). Building on these foundational approaches, the field has seen novel data formats such as the Stanford Alpaca (Taori et al. [39]), complementing projects like LLaMA, and datasets akin to Orca (Mukherjee et al. [29]), which re-envision FLAN through the capabilities of LLMs. Amidst these advancements, the LIMA (Zhou et al. [49]) study emerged, emphasizing the importance of high-quality datasets, inspiring the Open-Platypus (Lee et al. [17]) dataset which synthesizes these varied approaches. This dataset, noted for its high caliber, has become an almost indispensable resource in the STEM fields and consistently ranks highly on the Hugging Face open LLM leaderboard, signaling its critical role in current research and applications.

These research endeavors and the inherent challenges they address underscore a concerted effort to craft high-quality datasets, reflecting the field's commitment to precision and robustness in data-driven models.

3 Methods

In this section, we delineate the methodologies employed in our challenge submission. Adhering to the guidelines of the challenge and taking cues from the current trends in benchmark testing and recent scholarly advancements, we crafted a bespoke dataset tailored to align with the specificities of the task. This dataset was meticulously designed to resonate with the LLaMa2 70B model, forming the bedrock of our training process. Furthermore, we harnessed the power of QLoRA Fine-tuning,

Dataset Name	Task	# Size	License Type
PRM800K: A Process Supervision Dataset [21]	Math QA	13k	MIT
ScienceQA: Science Question Answering [26]	Science QA	1.3k	Creative Commons Attribution-NonCommercial-ShareAlike 4.0
ReClor: A Reading Comprehension Dataset Requiring Logical Reasoning [47]	Logical reasoning QA	4.5k	Non-commercial
TheoremQA: A Theorem-driven Question Answering Dataset [2]	STEM QA	0.5k	MIT
tigerbot-kaggle-leetcodesolutions-en-2k [40]	Coding problem solving	0.4k	Apache-2.0
ARB: Advanced Reasoning Benchmark for Large Language Models [36]	STEM QA	0.7k	MIT
Openassistant-guanaco [10]	Conversation	0.8k	Apache-2.0
Multi-News [12]	Summarization	4k	Other
FaithDial [11]	Multi-turn conversations	4.5k	MIT
LIMA: Less Is More for Alignment [49]	Conversations	1k	CC BY-NC-SA
Codegen [38]	Coding problem solving	4k	
Dolly [6]	Creative Writing, Closed QA, Open QA, Summarization, Information Extraction, Classification, Brainstorming	15k	cc-by-sa-3.0

Table 1: Datasets, Tasks, Number of Data Size, and Licenses. Our datasets, primarily showcased under Open-Platypus [17], demonstrate a strong emphasis on STEM and logical reasoning. In addition to these specialized datasets, we have also incorporated a diverse range of tasks such as general QA and conversation. This approach is part of our strategic effort to conduct instruction tuning across various fields, ensuring a comprehensive and robust training for diverse applications.

a cutting-edge technique, coupled with advanced attention mechanisms, to optimize the model's performance. These collective methodologies embody our strategic approach to addressing the challenge's complexities.

3.1 Custom Dataset

The foundation of our methodology is predicated on the creation of a custom dataset, meticulously aligned with the challenge's emphasis on open-source materials and benchmark tests encompassing logic reasoning types of multiple-choice QA scenarios and conversational chat tasks. This alignment necessitated the selection and composition of a dataset that adheres strictly to open-source principles, specifically excluding any datasets generated by LLMs themselves due to the challenge's guidelines.

We used the Open-Platypus[17] dataset as the reference dataset. Our choice of the Open-Platypus was driven by its reputation as a high-quality, open-source dataset in the STEM field, as evidenced by its frequent use among top performers on the HuggingFace Open LLM Leaderboard. In line with the recommendations from the [49], our focus was on maintaining high quality within custom dataset. In this case, we used the method utilized in the platypus's data curation. We utilized Sentence Transformer's embeddings (Reimers and Gurevych [35]) to assess the cosine similarity between sentences, aiming to curate a dataset of the highest calibre. To compensate for the absence of approximately 4k coding-related data excluded due to LLM origination, we integrated selections from the codeGen[38] dataset, ensuring diversity and richness in our data.

In our quest to fine-tune the model effectively within the 24-hour constraint, we also incorporated datasets covering diverse tasks. Drawing inspiration from [5], we observed that diverse instruction tuning significantly enhances performance. We began customizing with the Alpaca format used in Platypus as a base, carefully adapting it to suit our specific requirements. The data composition underwent rigorous testing across eight different configurations, incorporating additional datasets like CoT[48] to experiment with various permutations. Ultimately, we established our dataset structure as outlined in **Table 1**. Furthermore, we implemented a strategy to identify and eliminate any training data with a cosine similarity exceeding 0.9, ensuring the uniqueness and quality of our dataset. The experimental results of these eight data compositions can be found in **Section 4 Experiment**.

This comprehensive approach to dataset creation and optimization not only adhered to the challenge's stipulations but also positioned us advantageously to leverage the full potential of our selected LLM, laying a solid foundation for the fine-tuning process that follows.

3.2 QLoRA Fine-tuning

In fine-tuning, our team utilized the huggingface's [27] library, further optimized and extended in custom-developed Axolotl library [31]. This enhanced library integrates a wide range of advanced functionalities such as LoRA, QLoRA, flash attention, xformers attention, and gptq, ensuring a comprehensive and versatile toolkit tailored to our specific requirements. Specifically, for this study, our team applied QLoRA technique in conjunction with flash attention. This combination was chosen due to the efficiency and effectiveness of flash attention in handling large-scale language models, particularly in contexts demanding high computational efficiency and low memory footprint.

Low-Rank Adaptation The concept of low-rank adaptation, introduced by Hu et al. [15], enables these models to efficiently adapt within a reduced subspace. We apply this concept by reparametrizing the update to the pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ as a low-rank decomposition: $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d,k)$. In this structure, W_0 remains static during training, while A and B are trainable, representing the adaptation.

For an input x, the layer's output in the original network is $h = W_0 x$. With low-rank adaptation, the forward pass is expressed as:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{1}$$

This reparametrization begins with A having a random Gaussian initialization and B set to zero, ensuring $\Delta W = BA$ initially equals zero. To ensure balanced learning, $\Delta W x$ is scaled by αr, with α being a constant proportional to r. This scaling negates the need for extensive hyperparameter retuning when r changes, facilitating a more streamlined adaptation process.

Quantization The adoption of quantization techniques was crucial for deploying the LLaMa2 70B model on a single A100 40GB GPU. While various quantization methods exist, ranging from 8-bit to 2-bit, we placed our trust in the QLoRA method, primarily due to its strong performance as reported in [10]. In our implementation, the following QLoRA configuration was used:

$$Y_{\mathrm{BF16}} = X^{\mathrm{BF16}}\mathrm{doubleDequant}(c_1^{\mathrm{FP32}}, c_2^{k-\mathrm{bit}}, W^{\mathrm{NF4}}) + X^{\mathrm{BF16}} L_1^{\mathrm{BF16}} L_2^{\mathrm{BF16}}, \tag{2}$$

where **doubleDequant(·)** is defined as:

$$\mathrm{doubleDequant}(c_1^{\mathrm{FP32}}, c_2^{k-\mathrm{bit}}, W^{k-\mathrm{bit}}) = \mathrm{dequant}(\mathrm{dequant}(c_1^{\mathrm{FP32}}, c_2^{k-\mathrm{bit}}), W^{\mathrm{4bit}}) = W^{\mathrm{BF16}}. \tag{3}$$

For parameter updates, gradients are computed only for LoRA parameters using 16-bit BrainFloat. This quantization configuration is detailed in **Table 2**. The equation (1) is introduced in the [15], while equations (2) and (3) are adopted from the [10].

QLoRA Configuration:

- Y_{BF16}: The output tensor in BF16 format, where BF16 refers to 16-bit BrainFloat representation.
- X_{BF16}: The input tensor, also in BF16 format.

- c_1^{FP32}, $c_2^{k-\text{bit}}$: Quantization constants used in the double quantization process. Notably, c_2 uses the FP8 format.
- L_1^{BF16}, L_2^{BF16}: Coefficients for additional linear transformations, represented in BF16 format.

Configuration Parameter	**Value**
quantization method	bitsandbytes
load in 8bit	False
load in 4bit	True
llm int8 threshold	6.0
llm int8 skip modules	None
llm int8 enable fp32 cpu offload	False
llm int8 has fp16 weight	False
bnb 4bit quantization type	nf4
bnb 4bit use double quantization	True
bnb 4bit compute dtype	bfloat16

Table 2: Bitsandbytes Quantization Config and Value.

Hyperparameters for QLoRA In setting up our model, taking cues from the Platypus [17] approach, we determined our learning rate and LoRA α values accordingly in **Table 3**. In our study, we conducted a hyperparameter search as introduced in the [10]. This search for LoRA included varying parameters such as LoRA dropout, LoRA rank with values in 4, 8, 16, 32, 64, and the configuration of LoRA layers, which included options like key+query, all attention layers, all FFN layers, all layers, and attention + FFN output layers. Our focus was on optimizing LoRA r values and layer combinations to maximize performance within limited resources. According to [10], LoRA r is unrelated to final performance if LoRA is used on all layers. we managed to reduce GPU usage by lowering the LoRA r values and discovered that targeting attention + FFN output layers, as opposed to all layers, maintained comparable performance levels. Additionally, to further minimize GPU consumption, we employed strategies such as using low bit optimizers, adjusting sequence lengths, and reducing batch sizes. While these approaches initially posed challenges in training speed, leveraging Flash attention 2 significantly improved this aspect.

Hyperparameters	**Llama2-70B**
batch size	4
micro batch size	1
num epochs	1
learning rate	3e-4
cutoff len	1024
lora r	4
lora α	16
lora dropout	0.05
lora target modules	attention + FFN output layer
train on inputs	False
add eos token	True
group by length	True
prompt template	alpaca
optimizer	paged adamw 8bit
lr scheduler	cosine
weight decay	0
warmup steps	100

Table 3: Hyperparameters for QLoRA with 70B Models.

Through these methods, we successfully fine-tuned the LLaMa2 70B model on a custom-dataset within **16 hours**, utilizing **a single A100 GPU with 39.56GB usage**.

4 Experiment

Our experiments were conducted in two major phases. Initially, we focused on fine-tuning large-scale language models on a single A100 40GB GPU, emphasizing efficient resource utilization. Following this, we delved into identifying the optimal dataset composition within these constraints, ensuring the most effective training under limited computational resources.

Models We used a range of models, including 30B and 65B LLaMA1, 13B and 70B LLaMA2 models, and the Mistral 7B. The majority of our GPU usage tests were primarily conducted on the 70B models. Upon discovering the feasibility of fine-tuning the 70B model within our resource limits, we embarked on tuning this model with eight different versions of datasets. In our comparative analysis of the final dataset versions across these five models, the 70B model consistently outperformed others in terms of performance. Consequently, we decided to use the 70B model for our final implementation.

GPU Usage Tests For our GPU usage experiments, we focused on optimizing the hyperparameters based on the default QLoRA configuration provided by the Axolotl library [31]. To expedite GPU testing, we utilized the alpaca 2k test dataset [28] for comparing loss metrics. This approach allowed us to quickly evaluate and refine our settings, ensuring optimal performance within our computational constraints. In section 4.2, you can compare detailed experimental results.

Benchmark Tests For our benchmark testing, we employed the HELM [20] system, focusing on a subset of the Massive Multitask Language Understanding (MMLU [14]), which includes 17 out of the 57 subjects, as well as BBQ [33], TruthfulQA [22], and CNN/DailyMail [37]. Each test consisted of 50 examples, and the decoding system adhered to the standard settings provided by the HELM test environment. This setup was in alignment with the challenge leaderboard evaluation system. Based on these outcomes, we finalized the model configuration for our challenge submission.

4.1 Resource Efficiency Optimization

LoRA Configuration We extensively explored the configuration of LoRA to optimize the model's performance while managing resource constraints. Keeping the LoRA α value fixed at 16, we investigated the impact of varying the rank parameter LoRA r. As illustrated in Table 4, changing the r value showed a clear correlation with GPU usage and training loss, highlighting the delicate balance between computational efficiency and model accuracy. Specifically, we found that reducing the r value from 64 to 4 led to a modest decrease in GPU usage (from 51.27 to 46.49), with only marginal differences in training loss. Consequently, for our fine-tuning process, we settled on a r value of 4, optimizing both resource usage and model performance.

Further investigations into LoRA target layer combinations, as shown in Table 5, revealed that targeting specific layers for adaptation significantly impacts GPU utilization, training loss, and overall training time. These layer combinations lead to reduced GPU usage and shorter training times, with the increase in training loss being negligible compared to the baseline (all layers). Notably, configurations like ‘attention + FFN output layers‘ demonstrated optimal balances between GPU usage and training efficiency, indicating their suitability for resource-limited settings.

LoRA r	4	8	16	64
GPU Usage	46.49	46.95	47.43	51.27
Train loss	0.822	0.824	0.825	0.825

Table 4: Impact of LoRA r Value on GPU Usage and Training Metrics. Unlike the hyperparameters in Table 2, this table reflects a scenario where the cutoff length is set to 4096 and the optimizer used is paged adamw 32bit.

Flash Attention 2 The integration of Flash Attention 2 in our QLoRA setup significantly influenced our model's computational performance and resource utilization, as Table 6 clearly demonstrates. When compared with the QLoRA-only approach, the integration of Flash Attention 2 resulted in a substantial reduction in GPU usage from 68.13 to 46.95 and nearly halved the training time from 36m 12s to 19m 18s, with only a negligible increase in training loss. It's to note that the experiment without Flash Attention 2 was unique to this particular setup, providing a clear contrast in performance metrics.

LoRA Layer Combination	GPU Usage	Train loss	Time
key+query	**41.39**	0.830	**22m 12s**
all attention layers	**41.57**	0.801	**22m 43s**
all FFN layers	41.72	**0.798**	23m 30s
all layers	42.09	**0.794**	24m 4s
attention + FFN output layers	**41.46**	**0.798**	**22m 30s**

Table 5: Impact of LoRA Layer Combinations on Training Metrics. The combinations of layers in this table were inspired by and executed based on methodologies from the [10]. This approach aimed at maximizing efficiency without compromising performance, allowing for a nuanced comparison of different layer combinations and their impact on training metrics.

Method	QLoRA + Flash Attention 2	QLoRA Only
GPU Usage	46.95	68.13
Train loss	0.824	0.820
Time	19m 18s	36m 12s

Table 6: Comparison of QLoRA with and without Flash Attention 2. This experiment was conducted under the conditions where LoRA r value was set at 8, as referenced in Table 4.

4.2 Data Composition for Instruction Tuning

Dataset Composition Regarding our strategy for dataset composition, we meticulously tested and analyzed the performance of different dataset combinations across various model versions, as detailed in Table 7. These combinations included various proportions of datasets like Dolly, Filtering Platypus, Codegen, FaithDial, Multi-News, Lima, and FLAN-v2-Cot. Filtering platypus refers to a dataset other than the one created by LLM in compliance with the challenge. A notable aspect of our experiment was the comparison between Version 2 and Version 3, which were identical in dataset composition but differed only in the number of training epochs. This contrast was implemented to examine the impact of increasing epochs from 1 to 2 on model performance.

	Dataset Combination
Version 1	Filtering Platypus
Version 2	Dolly(100%) + Filtering Platypus + Codegen(10%) + Faithdial(20%) + Multi-News(10%)
Version 3	Dolly(100%) + Filtering Platypus + Codegen(10%) + Faithdial(20%) + Multi-News(10%)
Version 4	Dolly(100%) + Filtering Platypus + Codegen(10%) + Faithdial(40%) + Multi-News(20%)
Version 5	Dolly(100%) + Filtering Platypus + Codegen(10%) + Faithdial(20%) + Multi-News(10%) + Lima(100%)
Version 6	Dolly(100%) + Filtering Platypus + Codegen(10%) + Lima(100%)
Version 7	Dolly(100%) + Filtering Platypus + Codegen(10%) + Lima(100%) + FLAN-v2-Cot(2k)

Table 7: Model Version of Dataset Combination. 'Filtering Platypus' mentioned in each version represents a combined dataset, which includes all data from rows 1 to 7 as listed in Table 1.

Our benchmark tests, summarized in Table 8, utilized a variety of datasets: MMLU (comprising 17 subjects), BBQ, TruthfulQA, and CNN/DailyMail. For CNN/DailyMail, we recorded five distinct metrics, listed in sequence. The results from these benchmarks were intriguing, revealing that Version 5, despite its intricate dataset combination, emerged as the most effective version. Notably, Version 5 not only showed robust performance across all metrics but also demonstrated a significant balance in performance across different types of tasks.

In the context of the challenge, scores were computed using a mean-win-rate [34] calculation, which essentially quantifies the frequency of a model outperforming others across various comparison scores. Based on this scoring methodology, our Model Version 5 achieved the highest results on the leaderboard in our models. Consequently, we selected Version 5 for our final submission, underlining its superior performance in balancing resource efficiency with high accuracy across a diverse range of benchmarks.

Model	MMLU	BBQ	TruthfulQA	CNN/DailyMail
Version 1	0.70	0.92	0.74	0.14, 0.67, 0.42, 0.38, 0.31
Version 2	0.71	0.92	0.76	0.16, 0.67, 0.45, 0.67, 0.19
Version 3	0.71	0.96	0.54	0.15, 0.67, 0.45, 0.44, 0.15
Version 4	0.72	0.84	0.72	0.17, 0.67, 0.45, 0.52, 0.19
Version 5	0.73	0.90	0.76	0.17, 0.67, 0.44, 0.42, 0.17
Version 6	0.69	0.96	0.7	0.15, 0.56, 0.36, 0.44, 0.22
Version 7	0.70	0.84	0.64	0.14, 0.67, 0.5, 0.44, 0.15

Table 8: Result of Benchmark Test.The testing environment for these benchmarks was identical to that used in the challenge leaderboards. For the MMLU scores, only 17 out of 57 categories were utilized. Additionally, all datasets were evaluated based on 50 examples each to calculate the values. The columns under CNN/DailyMail represent ROUGE-2, Stereotypes (race), Stereotypes (gender), Representation (race), and Representation (gender), respectively.

Concluding Observations Our comprehensive experimentation culminated in achieving an efficient fine-tuning process for the LLaMA2 70B model. Utilizing Dataset Version 5, we successfully optimized the model to operate within the constraints of a single GPU, showing a total GPU usage of only 39.56 GB. Remarkably, we managed to train the model with a mere 0.0247% of trainable parameters, completing the process in just 17 hours. This result highlights the effectiveness of our fine-tuning strategy in balancing computational resources and model performance.

Model	Dataset	GPU Usage	Trainable Parameter	Training Time
LLaMA2 70B	Version 5	39.56	0.0247% (17M)	17h

Table 9: Final Result. This table represents our final training strategy. The actual number of trainable parameters involved in this setup is approximately 17 million.

5 Conclusion

Efficiency Our team's innovative approach to fine-tuning the Llama 2 70B parameter model has demonstrated remarkable efficiency. Leveraging the power of a single A100 GPU with 40GB of GPU memory, we successfully completed the task within a 24-hour (1-day) time frame. This feat was achieved through the strategic application of several advanced techniques, including QLoRA, Flash Attention 2, and multiple dataset curation. Given the same amount of train data volume, this task would typically require 8 A100 GPUs with 80GB. This needs a much longer time frame if none of these strategies were used.

Performance In the challenge ranking, our performance was evaluated through a two-stage process: open evaluation and secret evaluation. The evaluations proceeded in the order of open followed by secret, each utilizing distinct datasets. In the open evaluation, among 33 participating teams, our team secured the 7th place. The secret evaluation was conducted only for the top half of the teams from the open evaluation, where we competed among 17 teams and achieved the 11th rank.

While we demonstrated strong performance in QA task during the open evaluation, our ranking in the secret evaluation was affected by some unforeseen issues. Notably, a few metrics unexpectedly recorded 'NULL' values, leading to a lower ranking in the secret evaluation. This discrepancy in the results suggests areas for further investigation and improvement in our model's robustness across diverse datasets.

Final Remarks This project underscores the potential of resource-efficient methodologies in achieving high-quality results in machine learning tasks. Our findings highlight the importance of innovative techniques like QLoRA and Flash Attention 2 in optimizing the use of computational resources while maintaining the performance. Moving forward, our focus will be on refining these methods to bolster our model's performance across a wider range of datasets and metrics, paving the way for more accessible and robust AI solutions in the future.

Acknowledgements

We would like to express our gratitude to the following individuals for their invaluable contributions to this research. We thank Park, Se Un for his thorough and insightful review of this technical report, which significantly enhanced the quality and clarity of our work.

References

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

[2] Chen, W., Yin, M., Ku, M., Lu, P., Wan, Y., Ma, X., Xu, J., Wang, X., and Xia, T. (2023a). Theoremqa: A theorem-driven question answering dataset.

[3] Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. (2023b). Longlora: Efficient fine-tuning of long-context large language models. *arXiv:2309.12307*.

[4] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

[5] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models.

[6] Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. (2023). Free dolly: Introducing the world's first truly open instruction-tuned llm.

[7] Dao, T. (2023). Flashattention-2: Faster attention with better parallelism and work partitioning.

[8] Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

[9] Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). Llm.int8(): 8-bit matrix multiplication for transformers at scale.

[10] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

[11] Dziri, N., Kamalloo, E., Milton, S., Zaiane, O., Yu, M., Ponti, E., and Reddy, S. (2022). Faithdial: A faithful benchmark for information-seeking dialogue. *arXiv preprint, arXiv:2204.10757*.

[12] Fabbri, A. R., Li, I., She, T., Li, S., and Radev, D. R. (2019). Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model.

[13] He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. (2021). Towards a unified view of parameter-efficient transfer learning. *CoRR*, abs/2110.04366.

[14] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *CoRR*, abs/2009.03300.

[15] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

[16] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.

[17] Lee, A. N., Hunter, C. J., and Ruiz, N. (2023). Platypus: Quick, cheap, and powerful refinement of llms.

[18] Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *CoRR*, abs/2104.08691.

[19] Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *CoRR*, abs/2101.00190.

[20] Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. (2023). Holistic evaluation of language models.

[21] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023). Let's verify step by step.

[22] Lin, S., Hilton, J., and Evans, O. (2021). Truthfulqa: Measuring how models mimic human falsehoods. *CoRR*, abs/2109.07958.

[23] Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

[24] Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., and Tang, J. (2021a). P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

[25] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030.

[26] Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. (2022). Learn to explain: Multimodal reasoning via thought chains for science question answering.

[27] Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. (2022). Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

[28] mhenrichsen (2023). codegen.

[29] Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. (2023). Orca: Progressive learning from complex explanation traces of gpt-4.

[30] Ni, J., Xue, F., Jain, K., Shah, M. H., Zheng, Z., and You, Y. (2023). Instruction in the wild: A user-based instruction dataset. `https://github.com/XueFuzhao/InstructionWild`.

[31] OpenAccess-AI-Collective (2023). axolotl. `https://github.com/OpenAccess-AI-Collective/axolotl`.

[32] OpenAI (2023). Gpt-4 technical report.

[33] Parrish, A., Chen, A., Nangia, N., Padmakumar, V., Phang, J., Thompson, J., Htut, P. M., and Bowman, S. R. (2021). BBQ: A hand-built bias benchmark for question answering. *CoRR*, abs/2110.08193.

[34] Perlitz, Y., Bandel, E., Gera, A., Arviv, O., Ein-Dor, L., Shnarch, E., Slonim, N., Shmueli-Scheuer, M., and Choshen, L. (2023). Efficient benchmarking (of language models).

[35] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

[36] Sawada, T., Paleka, D., Havrilla, A., Tadepalli, P., Vidas, P., Kranias, A., Nay, J. J., Gupta, K., and Komatsuzaki, A. (2023). Arb: Advanced reasoning benchmark for large language models.

[37] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

[38] skar02 (2023). codegen.

[39] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

[40] TigerResearch (2023). tigerbot-kaggle-leetcodesolutions-en-2k.

[41] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

[42] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models.

[43] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[44] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2022). Self-instruct: Aligning language model with self generated instructions.

[45] Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

[46] Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., and Tian, Q. (2023). Qa-lora: Quantization-aware low-rank adaptation of large language models.

[47] Yu, W., Jiang, Z., Dong, Y., and Feng, J. (2020). Reclor: A reading comprehension dataset requiring logical reasoning. *CoRR*, abs/2002.04326.

[48] Zhang, Z., Zhang, A., Li, M., and Smola, A. (2022). Automatic chain of thought prompting in large language models.

[49] Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., and Levy, O. (2023). Lima: Less is more for alignment.