Fine-tuning Google's Gemma (7b) LLM
Context Crafters: CS 7643

James Mungai*, Krittaprot Tangkittikun*, Kevin Kori*
Georgia Institute of Technology
REDACTED@gatech.edu

Abstract

Since the launch of ChatGPT[1] in late 2022, the application of large language models (LLMs) for question-answering has grown significantly, attributed to their remarkable ability to answer general questions. Our project aims to enhance the user experience on the Kaggle platform by leveraging a pre-trained transformer model to provide prompt and relevant responses to user inquiries. This approach seeks to reduce users' effort in searching through the website or contacting support staff. A key challenge is that out-of-the-box LLMs are not specifically trained on niche domains like Kaggle and may not possess the requisite domain-specific knowledge. To address this, we employed a quantized low-rank adaptation (QLoRA) technique to fine-tune the Google Open-source Gemma LLM, which has 7 billion parameters, on a single GPU with a custom Kaggle-specific synthetic dataset generated from scraped data. Our findings indicate that while the fine-tuned model successfully acquired domain-specific knowledge from the training data, it still occasionally produces hallucinations and may revert to its pre-trained general knowledge when faced with insufficiently specific questions and contexts.

Our implementation can be found on the Github Repo: kaggle-gemma-peft.

1. Introduction/Background/Motivation

Kaggle[2] is an online community for data science and machine learning (ML) that serves as a learning platform for both novices and seasoned professionals, offering realistic practice problems to sharpen data science skills. Currently, navigating its extensive documentation or forums can be daunting for many users, potentially limiting their ability to make the most of the platform. This project aims to address this challenge by developing an artificial intelligence (AI) assistant specifically tailored to streamline the process of extracting information from Kaggle's documentation. The Question Answering Assistant is designed to significantly improve user experience and reduce the workload of platform staff by providing immediate and accurate responses to user inquiries.

We expect to build such an AI assistant with the open-source pre-trained large language model Gemma-7B, a recently released open-source language model by Google that was claimed to outperform other well-known models (e.g., Llama and Mistral), when matched for parameter size, on standard benchmarks such as MMLU, PIQA, and SIQA [5]. The disadvantage of the pre-trained language model is that it might not possess the requisite domain-specific knowledge related to the Kaggle platform.

To resolve this weakness, the model needs to be fine-tuned with a Kaggle-specific synthetic dataset generated from data scraped from kaggle.com/docs, instead of relying on established QA datasets such as SQuAD [3], to learn domain-specific knowledge.

Because of limited GPU resources, we aim to utilize parameter-efficient fine-tuning techniques such as quantized low-rank adaptation (QLoRA) so that fine-tuning can be performed on a single GPU (e.g., V100 or RTX3080TI) with minimal impact on the model performance when compared to full fine-tuning.

2. Dataset Preparation

Our project involved formatting Kaggle's "How To Use Kaggle"[3] web documentation for language model training, using Selenium and Beautiful Soup for scraping, and manually ensuring data accuracy. The scraped data is then split into smaller chunks where we applied techniques

[1] chat.openai.com
[2] kaggle.com
[3] kaggle.com/docs

frameworks like Self-Instruct Framework [6], Bonito [2], and Alpaca-Cleaned [4] to develop training and testing instruction-finetuning dataset with Gemini Pro 1.5 [4], supplemented by GPT 3.5 and 4 [5] for 20% of the data. Table 1 below contains the summarized dataset datasheet for reference.

The dataset can also be found on the Github Repo: kaggle-gemma-peft/dataset.

2.1. Defining the Instruction Data

Following the approach by [6], we produced a collection of instructional data consisting of various instructions I_t, where each I_t articulates a question or task using natural language. Each task t is paired with at least one input-output combination, denoted as $(X_{t,i}, Y_{t,i})$ for i ranging from 1 to n_t. We expected that our refined model M would generate the output $Y_{t,i}$ when given the instruction I_t alongside its respective input $X_{t,i}$, for all values of i from 1 to n_t.

2.2. Dataset Format

The datasets are in the form of *input*, *instruction*, *output* [6], where:

- **input** is the provided context extracted from the kaggle.com/docs website content.
- **instruction** is the question given to the model.
- **output** is the answer expected from the model.
- The dataset contains diverse Q&A tasks like answering true-false, yes-no, extractive question answering, and natural language inference of possible user questions.

For more details on the dataset, please refer to the complete datasheet available in the Appendix section.

3. Approach

In this study, we utilize our synthetic Kaggle docs dataset formatted in a "Self-Instruct" [6] like approach, an almost annotation-free method for aligning pre-trained language models with instructions. We used a pre-trained Gemma(7b) transformer model from the Hugging-Face[6] Transformer library [7] with QLoRA to build a chatbot for Kaggle Question Answering tasks.

QLoRA [1] is a fine-tuning method that efficiently reduces memory usage by backpropagating gradients through a 4-bit quantized pre-trained language model into Low-Rank Adapters (LoRA). It incorporates innovations such as 4-bit NormalFloat (NF4), an optimal data type for normally distributed weights, Double Quantization to lower the memory footprint by quantizing constants, and Paged Optimizers to control memory spikes, ensuring memory savings without performance loss.

The data used in this experiment were tokenized using a Gemma tokenizer to prepare the inputs for the model. Our approach also utilized the BitsAndBytes Config[7] class to establish a model quantization configuration that enhances computational efficiency with reduced precision arithmetic. Like [1], we employed a 4-bit quantization for inference (load_4bit =True) using the 'nf4' type, to save on memory without reducing the model performance. This setup also allowed for double precision in the quantization process to improve accuracy. Please refer to the Appendix Section 6.3 for details on "Public Code Used".

3.1. Incremental Training

Gradient check-pointing is a critical mechanism that enhances memory management during incremental training. To implement this, the prepare_model_for_kbit_training function was employed to configure the model appropriately for incremental training. This function enabled gradient checkpointing, thereby optimizing memory utilization, and enhancing the efficiency of the training process.

In our configuration for Low-Rank Adaptation (LoRA), several hyperparameters are specifically tuned to optimize the model's performance in causal language modeling tasks. This setup involves the application of LoRA regularization to designated projection layers, providing precise control over the model's adaptive behavior. Recent findings by [1] highlight the significance of certain LoRA hyperparameters, notably the total number of LoRA adapters employed. Their research suggests that incorporating LoRA adapters across all linear layers within transformer blocks is essential to achieve performance on par with complete fine-tuning. Based on these insights, it is recommended to implement LoRA across all linear transformer layers to mirror the effectiveness of full model fine-tuning.

Given the relatively small size of our dataset, it was necessary to employ high values for both LoRA rank and alpha in our model configuration. Through systematic hyperparameter tuning, we determined that a rank value of 128 yielded the most effective performance for our specific task.

The summary of the implementation issues and fixes is provided in Table 2

[4] ai.google.dev
[5] platform.openai.com
[6] huggingface.co
[7] huggingface.co/docs/transformers/en/main_classes/quantization

	Synthetic Datasets		
	Kaggle_docs_QA_v1	**Kaggle_docs_QA_v2**	**Kaggle_docs_QA_v3**
Input field optional	Yes	No	No
# of Instances	564	793	1663
# of instances with empty inputs	272	0	0
ave. instruction length (in words)	14.45	16.37	15.42
ave. non-empty input length (in words)	15.79	58.19	66.34
ave. output length (in words)	21.64	5.14	22.82
Yes-No QA	14	16	124
True-False QA	0	385	598
Extractive QA and NLI	550	392	941
Used for	Initial model setup	Test and Validation set	Training set

Table 1. Synthetic dataset datasheet (summarized)

Topic	**Details**	**Resolution**
Environment Set-up	Bit and Bytes is only compatible with Linux OS.	Used Google Collab or Docker Container.
Memory Issue	Our model is too large to fit into the VRAM.	Gradient checkpoint Quantized Model Model Complexity (e.g., Alpha/Rank)
Padding	Using right-side padding on a decoder-only architecture model does degrade performance.	Choose padding side as left.
Storage Issue	The model size is large, saving one adapter can take 1.6 GB.	Only save the best model to manage the storage.
Data Formatting	Scraped data contains unwanted symbols, tags, or spaces.	Cleaned the data using regex and Python.

Table 2. Issues and fixes

4. Experiments and Results

Our empirical analysis evaluated various hyperparameter settings to assess their impact on the training and validation performance of the model. LoRA was applied to all linear transformer block layers, achieving full fine-tuning performance, as suggested by [1]. NF4 with double quantization and the bf16 computation datatype were employed, replicating the approaches of the original study to minimize GPU memory usage.

The rank of LoRA in these adaptations ranged from 256 to 64, indicating a shift from higher to lower model complexity. The LoRA alpha parameter varied from 512 to 32, representing a scaling factor from high to low. We maintained a consistent dropout rate of 0.1 and a weight decay of 0.001 across all LoRA layers in all trials to prevent overfitting, except in experiment #7, where the dropout rate was increased to 0.15, which increased the regularization effect.

Initially, a smaller effective batch size of 2×4 was used, which was later adjusted to 8×16 to expedite fine-tuning. This adjustment allowed for quicker training at the cost of reduced regularization, yet it maintained higher training stability while maintaining the memory footprint within the available resources.

A learning rate of 2e-4 was consistently applied, coupled with a cosine scheduler and warm-up ratio of 0.3. This setup was designed to gradually increase the learning rate at the start and then decrease it in later epochs to ensure a stable convergence to the optimal loss. The max_grad_norm was capped at 0.3 to address the issue of exploding gradients. The padding_side was configured to the left to align with the decoder-only architecture of the Gemma model to optimize the performance. Additionally, a Paged Optimizer was utilized throughout the experiments to manage memory spikes.

Experiments	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8 (Chosen)**
lora_rank	256	128	64	128	128	128	128	128
lora_alpha	512	256	32	32	32	256	256	256
target_modules	All	All	All	All	All	All	All	All
lora_dropout	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.15
bias	None	None	None	None	None	None	None	None
Batch Size	2	2	2	2	8	8	8	8
Gradient Accumulation Steps	4	4	4	4	16	16	16	16
Learning Rate	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
num_epochs	3	3	3	3	3	3	3	3
max_grad_norm	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
warmup_ratio	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
weight_decay	0.001	0.001	0.001	0.001	0.001	0.001	0.01	0.01
max_seq_length	512	512	512	512	512	512	512	512
Scheduler type	cosine	cosine	cosine	cosine	cosine	cosine	cosine	cosine
padding_side	left	left	left	left	left	left	left	left
GPU	V100	V100	V100	RTX 3080TI	RTX 3080TI	RTX 3080TI	RTX 3080TI	RTX 3080TI
Training Duration (mins)	67	46	37	60	32	31	30	30
Training Loss	0.48	0.41	0.49	0.5	1.5	1.03	1	1
Testing Loss	3.02	2.67	3.1	3.12	2.84	2.8	2.79	2.77

Table 3. Hyperparameter tuning results

4.1. Experiments Findings

Based on experiments 1, 2, and 3, see Table 3, it was observed that a LoRA rank of 128 yielded the best generalization, where the testing loss was the lowest at 2.67. In Experiments 4 and 5, a LoRA rank of 128 was maintained, while the alpha value was lowered to 32.

Empirically, it appears that an alpha of 32 caused underfitting, as both the training and testing losses increased from 2.67 in experiment 2 to approximately 2.8. In Experiments 7 and 8, the alpha was reverted to 256, and attempts were made to increase the weight decay and LoRA dropout to 0.01 and 0.15 respectively. This regularization adjustment slightly improved the training and testing losses to values that were not significantly different from those observed in Experiment 2, with the testing loss decreasing to 2.77.

As observed in Figure 1, the testing loss stagnated with no further improvement after two epochs (24 steps). Consequently, the experiments were concluded, and the fine-tuned model from experiment #8 was selected for further evaluation.

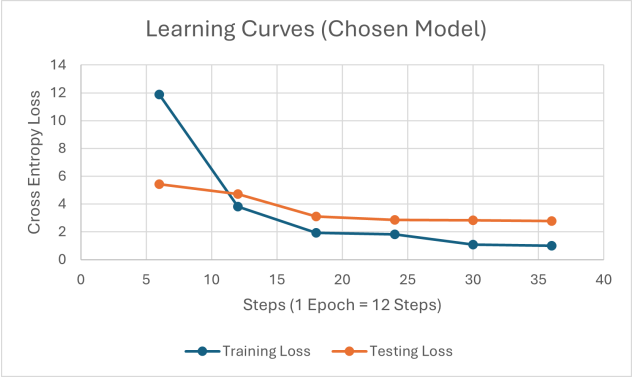

Figure 1. Learning Curves for the Chosen Model

4.2. Evaluation

In terms of model output evaluation, we originally planned use metrics such as BLEU, ROGUE, and Semantic Similarity. However, we discovered that performing inference on the full dataset is very time-consuming (e.g. greater than 1 h even on the GPU) and because the model output is much longer than the expected answer, it would be challenging to compare them word by word. Therefore, in the results section, we randomly sampled a handful of example questions to run inferences on and perform qualitative analysis.

4.2.1 Model Output Analysis

The Table 4 below shows some examples of hand-selected questions from the testing dataset in which the fine-tuned model answer is provided side by side with the expected answer (from the dataset). For specific questions, such as No. 1, 3, 4, 5, 6, and 7, the model seems to provide a reasonable and correct answer, sometimes with a minor incorrect reference. For No.8, we can observe that the model did hallucinate and tell the user that it is possible to clone a competition, even though the expected answer states that the user needs to set up the competition from the beginning.

Overall, we observed that the model had learned something from the training dataset, given that it had demonstrated the ability to answer some of the Kaggle-specific questions shown above and the learning curves observed during fine-tuning. However, we do not consider the output given from the model to be reliable enough to be launched to production, at least without a reinforcement learning feedback system in which the model can learn to behave in a way that real users find useful.

In future work, it would be useful to explore a more scalable method to generate larger and higher quality synthetic datasets and evaluate the model output using potentially more capable LLM (e.g., teacher models such as GPT-4 or Llama3) and not just human evaluators alone.

References

[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3

[2] Nihal V Nayak, Yiyang Nan, Avi Trost, and Stephen H Bach. Learning to generate instruction tuning datasets for zero-shot task adaptation. *arXiv preprint arXiv:2402.18334*, 2024. 2, 7

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 1

[4] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023. 2

[5] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 1

[6] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022. 2, 7

[7] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 2

No.	Question	Answer from Dataset	Answer from Fine-Tuned Model
1	What is the main objective of Playground competitions on Kaggle?	Playground competitions on Kaggle offer a "for fun" experience with relatively simple machine learning tasks.	Playground competitions on Kaggle are designed to provide a fun and accessible way for new users to learn the basics of machine learning and familiarize themselves with the Kaggle platform. They offer a low-pressure environment for learning and experimentation, helping users build foundational skills and confidence before tackling larger competitions. Reference: Official documentation on Playground competitions on Kaggle.
2	What is the typical format for most Kaggle competitions?	The typical format for most Kaggle competitions is the 'Simple' or 'classic' format.	Most Kaggle competitions are structured as binary classification problems, with participants making predictions on the target variable, which is typically binary in nature. The provided reference also highlights this point.
3	Where can users find a list of active Kaggle competitions?	Users can find a list of active Kaggle competitions on the Competitions listing page.	Users can find a list of active Kaggle competitions on the Competitions page, which is located at the URL provided in the text. This page provides a comprehensive overview of active competitions and allows users to filter and search for specific competitions based on their interests. Refer to the provided text for the exact URL.
4	Where should participants go if they have questions about a specific competition?	Participants should go to the competition's forums if they have questions.	Participants should go to the Discussion forum for the specific competition to ask their questions. This is where the community gathers to discuss and help each other with competition-related issues. Refer to the 'Discussion Forums' section of the provided text for more information.
5	What is one limitation on merging teams in Kaggle competitions?	One limitation is that most competitions have a team merger deadline, after which no mergers are allowed.	Teams cannot merge if the combined team has already submitted a solution file. This limitation is in place to ensure that teams cannot gain an unfair advantage by merging after the deadline or by taking advantage of the submission process to gain an advantage. Reference: Competition Rules and Regulations (Section 4.2)
6	Who can view the organization profile while it is in a pending state?	Non-members cannot view it	Only the organization members can view the organization profile while it is in a pending state. This information is available in the documentation on creating an organization.
7	True or False: The TPU performance monitor provides insights into hardware utilization and idle time, aiding in optimizing data pipelines for efficient TPU usage.	True	True. The TPU performance monitor indeed provides insights into hardware utilization and idle time, which are valuable for optimizing data pipelines for efficient TPU usage. This information is essential for maximizing TPU performance and efficiency. Reference: TPU Performance Monitor - Google Cloud AI Notebooks (Section: Monitoring and Optimization).
8	I want to run the same competition again, do I need to start from scratch?	For now, you are not able to clone a competition. You will need to start the set up from the beginning.	No, competitions can be rerun using the "Host >All Submissions" section of the winning notebook. This section allows hosts to easily launch a new competition based on the existing competition's details. Refer to the documentation on "Creating a New Competition" for more information.

Table 4. Samples Model Output Comparison with Dataset Answer

5. Appendix

5.1. Datasheet for Datasets

Below are the complete datasheets for our Kaggle documentation synthetic datasets. 5. We also provide known issues with the datasets. 6

5.2. Prompt Instructions

Similar to [6], we wrote a new prompt for each dataset that contained the explicit requirements for instruction generation using Gemini Pro 1.5. Each dataset had slightly different prompt instructions based on the desired objective.

- For the first dataset, *Kaggle_docs_QA_v1*, we gave the least restrictive instructions and generated at least 30 diverse Q&A pairs based on task t instructions. The input field is optional for this dataset.
- For the second dataset, *Kaggle_docs_QA_v2*, we gave specific instructions for the task t generation making it mandatory to ensure the entire data context is captured. We also provided instructions on the task and the output word length.
- For the final dataset, *Kaggle_docs_QA_v3*, we aimed to improve the quality and quantity of generated samples.

5.3. Public Code Used

In our dataset generation, we referred to the output of the quantized Bonito model for ideas on how to format our synthetic Q&A datasets from the scrapped raw data from Kaggle docs.

Like [2], We proceeded to refer to [6] for task generation as discussed in 7.2.

Bonito code link can be found here and the Self-Instruct Code and data are available at here.

In our model setup, configuration, and fine-tuning we made changes to LLM Prompt Recovery with Gemma Kaggle notebook by Aisuko.

- Implemented data cleaning to remove unwanted characters, html/markdown, and links.
- Implemented data preprocessing to format our dataset as expected (content, question, and answer).
- Implemented lookup capability to identify manipulable lora layers to be used in the fine-tuning.
- Implemented fast tokenizer version with a left sided padding at tokenizer setup.
- Used Causal LM task type which is great for question-and-answer tasks at lora configuration.
- Tuned the lora hyper-parameters and QLoRa hyper-parameters to suit our use case. We included techniques like gradient check pointing, early stopping, max grad norm, warmup, weight decay to attain good model performance.
- Implemented inference generation from fine-tuned model for validation exercise to measure the performance of our fine-tuned model.

Aisuko's Kaggle notebook link can be found here.

	Datasheet for datasets
Data source	https://www.kaggle.com/docs
Licenses and Privacy	https://www.kaggle.com/privacy
Scrapping tool	Selenium and Beautifulsoup
Date scrapped	March 18, 2024
Dataset format and structure	JSON {"input": 'str' context or input for the task, "instruction": 'str' describes the task to be performed, "output": 'str' the answer to the instruction}
Purpose	Question answering task for Kaggle.com-specific documentation including information about competitions, datasets, models, notebooks, compute, and more
Created by	Context crafters (Gatech students, CS7643, Spring 2024)
Question and answer pairs generating tool	Gemini Pro 1.5, GPT-3.5, GPT 4.0, Bonito

Table 5. Datasheet for dataset

	Synthetic Datasets		
	Kaggle_docs_QA_v1	**Kaggle_docs_QA_v2**	**Kaggle_docs_QA_v3**
Input field optional	Yes	No	No
# of Instances	564	793	1663
# of instances with empty inputs	272	0	0
ave. instruction length (in words)	14.45	16.37	15.42
ave. non-empty input length (in words)	15.79	58.19	66.34
ave. output length (in words)	21.64	5.14	22.82
Yes-No QA	14	16	124
True-False QA	0	385	598
Extractive QA and NLI	550	392	941
Used for	Initial model setup	Test and Validation set	Training set
Known issues	**Incomplete context**: Optional or summarized input/context, **Translated text:** some questions generated are for translating English to non-English words (primarily Spanish words)	**Short outputs**: One-word or short outputs Limited diversity for question-answer pairs per provided context	

Table 6. Known issues with the datasets