Optimizing Small Language Models with Task-Specific LoRA Adapters for Superior Performance in Memory-Constrained Environments

Anweasha Saha, Bharati Jagdish Panigrahi, Lance Garrick Soares,
Shaurya Bhatnagar, Vijay Ravichander

{anweasha, bpanigra, soaresl, sb25626, vijayrav}@usc.edu

December 17, 2024

Abstract

General-purpose LLMs are pre-trained on vast and diverse datasets, designed to handle a wide range of tasks, however, they might lack the specificity needed for highly specialized domains (Paul, September 29, 2023). On the other hand, hosting multiple specialized LLMs on the same hardware requires large computational resources making it infeasible. This report explores the use of Small Language Models (SLMs) with task-specific Low-Rank Adaptation (LoRA) adapters (VLLM, 2024) to address these limitations. By fine-tuning SLMs for tasks like code generation, mathematical reasoning, and text summarization, and dynamically swapping LoRA adapters, we achieve efficient, task-optimized performance. Evaluations on benchmarks such as HumanEval and GSM8K demonstrate that this approach enables SLMs to rival LLMs in specialized tasks while maintaining memory efficiency, offering a scalable solution for resource-constrained AI applications.

1 Introduction

The growing reliance on general-purpose Large Language Models (LLMs) has highlighted a significant trade-off between their broad applicability and the specialized performance required in niche domains. While LLMs are pre-trained on vast datasets to address diverse tasks, their performance may fall short in highly specialized scenarios. Furthermore, deploying multiple LLMs tailored to specific tasks is computationally prohibitive, especially in memory-constrained environments such as mobile devices and laptops.

To address these challenges, this project explores the potential of Small Language Models (SLMs) equipped with task-specific Low-Rank Adaptation (LoRA) adapters. By leveraging LoRA's efficiency, we aim to fine-tune SLMs for distinct tasks such as code generation, mathematical reasoning, text summarization, and paraphrasing. This method utilizes LoRA adapters for hot-swapping task-specific parameters, enabling adaptable and memory-efficient solutions without compromising performance.

Our approach involves selecting base SLMs like Qwen2.5-1.5B-Instruct and Llama-3.2 models, known for their efficiency and scalability. Task-specific LoRA adapters are trained on carefully curated datasets and integrated into these models, optimizing for both computational efficiency and task accuracy. Through rigorous evaluation metrics like HumanEval, GSM8K, and Hellaswag, this study seeks to demonstrate that task-specific adaptations of SLMs can achieve comparable or superior performance to general-purpose LLMs while adhering to memory constraints.

This research represents a step toward making AI more accessible and effective in resource-limited settings, providing a pathway for efficient task specialization without the need for massive computational resources.

2 Related Work

Recent studies highlight Low-Rank Adaptation (LoRA) as a memory-efficient fine-tuning method for large language models, preserving base model knowledge while minimizing "catastrophic forgetting." In LoRA Learns Less and Forgets Less (Biderman et al., 2024), LoRA's balance of learning and retention is analyzed, showing it maintains task diversity and prevents performance decline on earlier tasks. Extending this, MTL-LoRA (Yang et al.,

2024) addresses task interference in multi-task scenarios by incorporating task-specific parameters, which enhance task separability and reduce cross-task conflicts, as shown through t-SNE visualizations. Meanwhile, QLoRA (Dettmers et al., 2023) introduces advanced quantization techniques for LoRA, making high-capacity models feasible on constrained devices without loss in performance. Our research builds on these qualitative techniques by experimenting with task-specific LoRA configurations and quantization strategies to improve adaptability and memory efficiency across diverse tasks.

LoRA-FA (Zhang et al., 2023) optimizes memory use during fine-tuning by freezing projection-down weights and updating only projection-up weights in each LoRA layer, reducing memory costs by up to 1.4 times. LoRA-XS (Bałazy et al., 2024) takes parameter efficiency further by inserting a small trainable matrix between frozen low-rank matrices created through Singular Value Decomposition, achieving over 100-fold parameter reduction for 7B models. These methods offer promising paths for parameter-efficient adaptation in memory-constrained environments, which our research aims to explore and extend.

While these studies focus primarily on optimizing the performance and memory efficiency of large models, such as 7B-parameter models, our work uniquely focuses on Small Language Models (SLMs) that have significantly fewer parameters, making them inherently more suitable for memory-constrained environments. Unlike LoRA-FA and LoRA-XS, which concentrate on weight-freezing or advanced decomposition techniques, our approach employs hot-swappable LoRA adapters tailored to specific tasks, enabling modularity and dynamic adaptability for specialized tasks.

Additionally, MTL-LoRA addresses cross-task conflicts in multi-task learning, which aligns with our work's emphasis on task separability. However, while MTL-LoRA was designed for large-scale LLMs, our study adapts this principle to SLMs, aiming for similar benefits with substantially lower computational requirements. QLoRA's focus on quantization techniques inspires our use of memory-efficient configurations, but our work further extends these methods by demonstrating that task-specific LoRA adapters can achieve comparable performance to general-purpose LLMs even with smaller, more resource-efficient models.

3 Methodology

In the subsequent sections 3.1, we present a detailed overview of the components selected to conduct our study and analysis.

3.1 Base Models

As mentioned, a core goal of this project is to evaluate the inference capabilities of Small Language Models (Schick and Schütze, 2021), with a much smaller parameter count than that of larger general-purpose language models, and then try to boost these capabilities using Performance Efficient Fine-tuning techniques.

Llama-3.2-1B-Instruct

The Llama-3.2-1B-Instruct is a compact language model fine-tuned for instruction-following tasks. With 1 billion parameters, it is optimized for efficiency and accuracy, making it suitable for applications requiring reliable responses while operating within resource-constrained environments. It excels in text generation, comprehension, and task-specific adaptability, while maintaining a balance between speed and performance.

3.2 Comparison Models

To evaluate the performance of our base and fine-tuned models, we selected comparison models that represent state-of-the-art instruction-tuned architectures. Specifically, we included Llama-3.2-3B-Instruct and Mistral-7B-Instruct-v0.3. This selection of models varying in size and number of parameters provides insights into the trade-offs between model size, efficiency, and task-specific adaptability.

Llama-3.2-3B-Instruct & Mistral-7B-Instruct

The Llama-3.2-1B-Instruct is a 3-billion-parameter instruction-tuned variant of the Llama model family which is optimized to improve performance on tasks requiring understanding, reasoning, and generative outputs while balancing efficiency and computational cost. Mistral-7B-Instruct-v0.3 is a larger, high-performance model with 7 billion parameters which has achieved state-of-the-art results on reasoning, comprehension, and generative tasks, making it a competitive choice for real-world applications.

3.3 Datasets

In this section, we outline the datasets selected for training specific capabilities in our model, focusing on code generation, mathematical reasoning, and text summarization. By leveraging these diverse datasets, we aim to enhance the model's performance across a variety of tasks and domains.

Code Generation:
For code generation, we utilized a subset of **14,500** Python-specific instruction-response pairs from the Alpaca dataset to train a LoRA adapter. This dataset, containing 52,000 total instructions and demonstrations generated by OpenAI's text-davinci-003 engine, was selectively filtered to focus on Python-related tasks. Through this targeted subset, we aim to instruction-tune the model for enhanced performance in Python-based task adherence.

Mathematical Reasoning:
To enhance mathematical reasoning, we leveraged a dataset, which contains 12.5k math problems across diverse topics, such as algebra, geometry, precalculus, and number theory. Each problem is categorized by difficulty levels (e.g., Level 3, Level 5) and includes detailed solutions. (LightEval, 2024)

Text Summarization and Paraphrasing:
For text summarization and paraphrasing tasks, The Databricks-dolly-15k dataset (Conover et al., 2023) was used , an open-source collection of instruction-following records created by Databricks. We focus specifically on data in the closed question answering and information extraction categories for our study.

3.4 Evaluation Bechmarks

3.4.1 HumanEval

For Code Eval The **openai/openai_humaneval** dataset released by OpenAI includes **164** programming problems with a function signature, docstring, body, and several unit tests, handwritten to ensure these problems are not included in the training set of code generation models (Chen et al., 2021). This benchmark evaluates models on ***pass@k*** metric which measures the functional correctness of code generated by the model when allowed k completions. For the purpose of this project, we have chosen the values of $k \in \{1, 10\}$.

3.4.2 GSM8K

GSM8K (Grade School Math 8K) (Cobbe et al., 2021) is a dataset of **8.5K** high-quality, linguistically diverse grade-school math word problems. For this study, a subset of GSM8K to evaluate the performance of the LLM on mathematical reasoning.

3.4.3 Hellaswag

Hellaswag is a challenge dataset composed of **39905** training, **10042** validation and **10003** testing samples curated via Adversarial Filtering (AF) (Zellers et al., 2019). In this study we focus on subset of *50/192* tasks and set *n_shots = 1* to evaluate our models' text summarization and paraphrasing capabilities.

4 Experiments

4.1 Baseline Evaluation

4.1.1 Text Summarization and Paraphrasing

To evaluate the Text Summarization and Paraphrasing capabilities we made use of two benchmarks: the HellaSwag evaluation benchmark (Zellers et al., 2019) and the CNN/DailyMail dataset (Nallapati et al., 2016). We assessed the commonsense reasoning and contextual understanding capabilities of our models to quantify their understanding of implicit relationships, causality, and context within the text while also imparting a better grasp of nuances and ambiguities of language.
For Hellaswag, the primary evaluation metric was accuracy, calculated as the proportion of completions matching the correct option when 5 completions were allowed.
Additionally, we evaluated summarization and paraphrasing capabilities of the models using a validation subset of the CNN/DailyMail dataset. The model received article prompts and generated corresponding summaries, which were then stored in the following format:

```
{"article_id":
    "CNN/DailyMail article ID",
 "article":
    "Full input article text",
 "generated_summary":
    "Model-generated summary",
 "reference_summary":
    "Ground-truth summary"}
```

The summarization metrics assessed were: ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) to measure n-gram overlap and sequence coherence between generated and reference summaries.

4.1.2 Code Generation

We evaluated code generation capabilities using 164 coding tasks from (Chen et al., 2021). Each model received a task prompt and generated completions in $k \in \{1, 10\}$ iterations per task. Outputs were preprocessed to ensure compatibility with the HumanEval framework, then stored in JSONL format for structured evaluation:

```
{"task_id":
    "HumanEval task ID",
"completion":
    "Generated completion text"}
```

The HumanEval module assessed functional correctness, providing *pass@1* and *pass@10* metrics along with detailed pass/fail results.

4.1.3 Mathematical Reasoning

We evaluated the Mathematical Reasoning ability of our model using the GSM8K dataset (Cobbe et al., 2021). GSM8K is a benchmark specifically designed to test models' problem-solving and arithmetic reasoning skills through grade school level math word problems.
Each problem from the GSM8K dataset was presented to the model as a prompt in natural language. The model was tasked with generating the solution to the problem. To evaluate reasoning, under iterative settings, we generated 5 completions for each problem and the model-generated solutions were parsed to extract the final numerical answer to ensure that minor variations in wording did not affect evaluation results.
To assess model performance, the Exact Match (EM) Accuracy: proportion of problems where the model's final answer exactly matches the ground truth numerical answer was calculated.

4.2 Fine Tuning and training LoRA Adapters

4.2.1 Code Generation

In this work, we fine-tuned Low-Rank Adaptation (LoRA) adapters for three distinct model architectures: Llama3.2 1B, Llama3.2 3B, and Qwen 2.5B. Each adapter was strategically integrated across specific layers within the models to maximize efficiency and performance gains. These targeted layers included the attention-related projections—namely, `"q_proj"`(query projection), `"k_proj"` (key projection), `"v_proj"`(value projection), and `"o_proj"` (output projection). Additionally, adapters were placed within other functional layers such as `"gate_proj"`, `"up_proj"`, and `"down_proj"` to further capture and leverage nuanced dependencies within each model.
To evaluate the performance of the adapters and optimize their configuration for each model, we tested different ranks, experimenting with values of 16, 32, 64, and 128. These varying rank levels allowed us to control the degree of low-rank approximation within each adapter, directly impacting the balance between computational efficiency and model expressiveness. Throughout all experiments, we maintained a constant `"lora_alpha"` of 16. It's worth noting that no additional bias terms were introduced during the fine-tuning process. This setup allowed us to analyze the isolated impact of LoRA fine-tuning on the models' performance across various tasks and configurations.

4.2.2 Text Summarization and Paraphrasing

For text summarization and paraphrasing tasks, we fine-tuned the Llama 3.1 1B Instruct model using a similar approach to that employed for code generation. The LoRA adapters were strategically integrated into three distinct configurations to analyze their impact on model performance.

First, LoRA layers were applied exclusively to the attention-related projections (`"q_proj"`, `"k_proj"`, `"v_proj"`, and `"o_proj"`). This setup focused on optimizing the model's ability to capture contextual relationships within the input sequence.

Second, we added LoRA layers to the Multi-Layer Perceptron (MLP) layers, specifically targeting `"gate_proj"`, `"up_proj"`, and `"down_proj"`. This configuration emphasized the enhancement of the model's capacity for nonlinear transformations and feature extraction.

Finally, LoRA adapters were applied across both the attention-related and MLP layers, enabling comprehensive fine-tuning of the model for improved generalization and task-specific performance.

To assess the effectiveness of these configurations, we experimented with varying LoRA ranks, testing values of 16, 32, 64, and 128 for each setup. This enabled us to systematically explore the trade-offs between computational efficiency and representational capacity.

4.2.3 Mathematical Reasoning

For mathematical reasoning tasks, we adopted a similar structure of parameter-efficient fine-tuning as used in the other tasks. However, since applying LoRA layers across all layers (Attention + MLP) emerged as the most effective configuration in both code generation and text summarization and paraphrasing, we focused solely on this approach for mathematical reasoning.

Our experiments primarily involved varying the ranks of the LoRA layers, testing different values to optimize performance while maintaining computational efficiency. This streamlined approach allowed us to concentrate on refining the model's capacity for mathematical reasoning, leveraging the proven benefits of comprehensive fine-tuning with LoRA adapters.

5 Results

5.1 Code Generation

Our experimental results with Llama-3.2-1B-Instruct indicate that fine-tuning small language models (SLMs) with task-specific LoRA adapters enhances their code generation capabilities. However, the performance improvement is not substantial enough to bring these models on par with larger models such as Llama-3.2-3B-Instruct or Mistral-7B-Instruct-v0.3. This suggests that the 1 billion parameters in Llama-3.2-1B-Instruct may be insufficient to capture the complex patterns and intricacies required for effective code generation.

In contrast, fine-tuning slightly larger models like Llama-3.2-3B-Instruct results in a significant performance boost, increasing accuracy from approximately **8.5%** to **21.6%**. This improvement brings the model's performance closer to that of larger models like Mistral-7B, while maintaining a favorable trade-off between computational efficiency and performance.

Additionally, our results emphasize the critical role of the LoRA adapter rank. Models with higher rank LoRA adapters (e.g., **r128**) consistently outperform those with lower rank adapters (e.g., **r32**). The increased rank allows for more expressive parameterizations, enabling the model to capture more granular patterns, thereby improving performance.

Model Name	**pass@1**	**pass@10**
Base Models		
Llama-3.2-1B-Instruct	**0.0018**	**0.0122**
Llama-3.2-3B-Instruct	**0.0852**	**0.2781**
Mistral-7B-Instruct-v0.3	**0.3231**	**0.4329**
Our Finetuned Models		
Llama-1B-Code-r32	0.0016	0.0641
Llama-1B-Code-r64	0.0023	0.0765
Llama-1B-Code-r128	**0.0038**	**0.1124**
Llama-3B-Code-r32	0.1729	0.2137
Llama-3B-Code-r64	0.1095	0.2857
Llama-3B-Code-r128	**0.2160**	**0.3654**

Table 1: Evaluation Scores on HumanEval

5.2 Text Summarization and Paraphrasing

Next, we conducted experiments with three distinct configurations of LoRA adapters: 1) applying LoRA layers exclusively to the attention projections, 2) applying LoRA layers exclusively to the multilayer perceptron (MLP) layers, and 3) applying LoRA layers to both the attention and MLP layers. These fine-tuned architectures were evaluated on the CNN/DailyMail and HellaSwag benchmark datasets, and the results are presented as follows:

Model Name	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Base Models			
Llama-3.2-1B-Instruct	**0.2274**	**0.0521**	**0.2165**
Llama-3.2-3B-Instruct	**0.2933**	**0.0793**	**0.2478**
Mistral-7B-Instruct-v0.3	**0.3742**	**0.1893**	**0.2618**
Our Finetuned Models			
Llama-1B-LoRA-MLP-R32	0.2467	0.1193	0.2238
Llama-1B-LoRA-MLP-R128	0.0023	0.0765	0.2238
Llama-1B-LoRA-Attn-R32	0.3139	0.1346	0.2691
Llama-1B-LoRA-Attn-R128	0.2919	0.0793	0.2368
Llama-1B-LoRA-ALL-R32	0.2536	0.1375	0.2616
Llama-1B-LoRA-ALL-R128	**0.3252**	**0.1518**	**0.2707**

Table 2: Evaluation Scores on CNN/Dailymail

Our findings indicate that the models with LoRA layers applied to both the attention and MLP layers—regardless of the specific rank of the LoRA adapters—consistently outperform the other configurations in terms of ROUGE-1, ROUGE-2, and ROUGE-L scores on the CNN/DailyMail dataset. The attention layers are responsible for capturing long-range dependencies and token relationships, while the MLP layers focus on feature transformations and internal model representations. By incorporating LoRA adapters into both the attention and MLP layers, the model gains increased flexibility and expressiveness, allowing it to more effectively adapt to the summarization task.

This conclusion is further supported by the accuracy improvements observed on the HellaSwag dataset. Specifically, applying LoRA adapters of rank **128** to both the attention and MLP layers of the Llama-3.2-1B-Instruct model significantly enhances its commonsense reasoning capabilities. Fine-tuning the Llama-1B model with this configuration improved its accuracy from **45%** to **52.75%**, surpassing the performance of the Llama-3B model (a 3 billion parameter model) and making it comparable to the much larger Mistral-7B model. Notably, our fine-tuned model achieves this performance with a fraction of the size and computational resources required by larger language models.

Model Name	**Accuracy**	**Acc - Norm**
Base Models		
Llama-3.2-1B-Instruct	**0.4507**	**0.6073**
Llama-3.2-3B-Instruct	**0.449**	**0.5991**
Mistral-7B-Instruct-v0.3	**0.5633**	**0.747**
Our Finetuned Models		
Llama-1B-LoRA-MLP-R32	0.4499	0.6006
Llama-1B-LoRA-MLP-R64	0.4578	0.6843
Llama-1B-LoRA-MLP-R128	0.4987	0.711
Llama-1B-LoRA-Attn-R32	0.4447	0.5969
Llama-1B-LoRA-Attn-R64	0.4546	0.6764
Llama-1B-LoRA-Attn-R128	0.5121	0.7198
Llama-1B-LoRA-All-R32	0.449	0.5991
Llama-1B-LoRA-All-R64	0.4603	0.629
Llama-1B-LoRA-All-R128	**0.5275**	**0.7165**

Table 3: Evaluation Scores on HellaSwag

5.3 Mathematical Reasoning

We employed a similar training and validation pipeline to fine-tune Llama-3.2-1B-Instruct for enhancing its mathematical reasoning abilities using the GSM8k dataset. In line with our previous findings, LoRA layers were applied to both the attention and multilayer perceptron (MLP) layers to achieve most optimal performance. Consistent with earlier results, models incorporating higher rank LoRA adapters **(rank 128)** outperformed those with lower rank adapters due to their increased capacity to model more complex relationships and dependencies within the data dealing with number manipulation, logical structures, and relationships between entities. Our fine-tuned model, equipped with a LoRA adapter of rank 128, demonstrated a significant improvement in Exact Match Accuracy, increasing from **32.9%** to **64.56%**, a performance level comparable to much larger models such as Llama-3.2-3B-Instruct (3 billion parameters) and Mistral-7B-Instruct-v0.3 (7 billion parameters).

Model Name	**Flexible Extract**	**Strict Match**
Base Models		
Llama-3.2-1B-Instruct	**0.329**	**0.329**
Llama-3.2-3B-Instruct	**0.6513**	**0.6429**
Mistral-7B-Instruct-v0.3	**0.6679**	**0.6261**
Our Finetuned Models		
Llama-1B-Math-R32	0.2411	0.2388
Llama-1B-Math-R64	0.4264	0.4118
Llama-1B-Math-R128	**0.6456**	**0.6403**

Table 4: Evaluation Scores on GSM8k

6 Conclusion

This study demonstrates the effectiveness of fine-tuning Small Language Models (SLMs) with task-specific Low-Rank Adaptation (LoRA) adapters to achieve competitive performance in specialized tasks while maintaining computational efficiency. By dynamically applying LoRA adapters to both the attention and multilayer perceptron (MLP) layers, we enable SLMs to capture intricate task-specific patterns without the need for additional parameters or excessive computational overhead.

Our findings highlight several key observations. First, while extremely small models, such as Llama-3.2-1B-Instruct, show some gains in performance after fine-tuning, their limited parameter capacity constrains their ability to handle complex tasks like code generation. In contrast, slightly larger models like Llama-3.2-3B-Instruct demonstrate significant improvements, achieving a favorable trade-off between computational cost and performance.

Second, the rank of the LoRA adapters plays a pivotal role in determining the fine-tuning effectiveness. Higher rank LoRA adapters (e.g., rank 128) consistently outperform lower rank configurations (e.g., rank 32) across various tasks, including code generation, text summarization, and mathematical reasoning. The increased rank provides greater expressiveness, enabling the models to better capture fine-grained relationships and patterns inherent in specialized datasets.

For text summarization, applying LoRA adapters to both attention and MLP layers enhances the model's ability to efficiently capture long-range dependencies and transform features, delivering the best results. In commonsense reasoning tasks (HellaSwag), fine-tuning with higher-rank LoRA adapters (rank 128) allows smaller models like Llama-3.2-1B-Instruct to rival larger models, such as Llama-3B and Mistral-7B, while significantly reducing computational costs.

The success of this approach extends to mathematical reasoning tasks, where our fine-tuned Llama-3.2-1B-Instruct model achieved a remarkable improvement in Exact Match Accuracy on the GSM8K dataset, rising from 32.9% to 64.56%. This level of performance rivals that of models with 3 to 7 billion parameters, demonstrating the potential of SLMs equipped with well-optimized LoRA adapters to compete with Large Language Models (LLMs).

In conclusion, this work showcases a scalable and resource-efficient method for enhancing the performance of SLMs in specialized tasks. By leveraging task-specific LoRA adapters, we demonstrate that smaller models can deliver performance comparable to much larger models across a range of benchmarks, including HumanEval, CNN/DailyMail, HellaSwag, and GSM8K. This approach opens the door for deploying high-performing, task-optimized models in resource-constrained environments, providing a practical solution for a wide range of AI applications.

7 Future Scope

While this study demonstrates the effectiveness of Small Language Models (SLMs) enhanced with task-specific LoRA adapters for specialized tasks, several directions remain for future exploration. First, integrating more advanced quantization techniques, such as those used in QLoRA, could further reduce memory consumption and make these models more viable for deployment on edge devices. Second, expanding the scope of tasks to include domains like natural language inference, document classification, and conversational AI can test the adaptability and scalability of LoRA-equipped SLMs. Additionally, investigating methods for automatically selecting and swapping LoRA adapters based on task detection could enhance model efficiency in multi-task environments.

Future work could also explore hybrid architectures that combine LoRA with other fine-tuning strategies, such as prompt tuning or adapters with dynamic rank adjustments, to achieve an optimal balance between performance and resource usage. Additional methods, such as Spectrum (Hartford et al., 2024), Model Soups (Wortsman et al., 2022), and Model Merging, may also be explored. Finally, real-world deployment and evaluation of these models on mobile or IoT devices will validate their practicality and impact in memory-constrained environments.

Another avenue of exploration for optimizing language models in memory constrained environments would be to study the scaling of inference-time computation in LLMs as discussed in (Snell et al., 2024)

References

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. Lora-xs: Low-rank adaptation with extremely small number of parameters.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Eric Hartford, Lucas Atkins, Fernando Fernandes Neto, and David Golchinfar. 2024. Spectrum: Targeted training on signal to noise ratio.

LightEval. 2024. Math: A dataset for mathematical reasoning.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Anwesha Paul. September 29, 2023. General purpose vs. customizable llms: Weighing in on the debate.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

VLLM. 2024. Dynamically serving lora adapters.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, Weizhu Chen, and Yunhai Tong. 2024. Mtl-lora: Low-rank adaptation for multi-task learning.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning.