“Comparative Analysis of LLaMA 3 & Gemma: Evaluating the Strengths and Weaknesses of Two SLMs”

Michael Bui, Daehee Kim
College of Engineering, California State University Long Beach
Long Beach, USA

Abstract

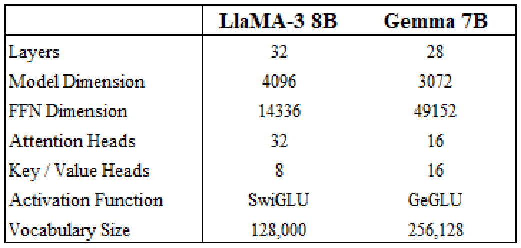
This study compares two SLMs, LLaMA-3 (8B) and Gemma (7B) on both high-end and low-end hardware. Performance was evaluated across logical and technical metrics (latency, resource usage) using benchmarking tests (MMLU, GSM-8K, ARC) as well as text generation prompts. Results showed that LLaMA consistently outperformed Gemma in speed and accuracy for most tasks, despite Gemma’s larger vocabulary and FFN dimensions. Findings suggest Meta’s LLaMA-3 iteration provides better performance and accuracy on lower-end devices than Gemma.

I. INTRODUCTION

The increasing advancement of artificial intelligence (AI) has increased the nuances in its usage and capabilities. This study aims to assess the performance and use cases of two SLMs, LLaMA-3 and Gemma, for future researchers and readers.

A. Background

Much of the world has adopted the usage of large language models (LLMs) today. However, a common misconception is that mainstream LLMs, such as OpenAI’s ChatGPT, are “a one size fits all” solution; this is incorrect. There are over thousands of various language models accessible on the web today that vary in purpose, as well as processing power. For example, certain models, such as LLaMA-3-8B (read as version 3, fed 8 billion parameters) and Gemma-7B (fed 7 billion parameters), are smaller in scale and take less parameters in training in order to cater to limited resources; these are known as small language models (SLMs). Additionally, these models can employ quantization to represent parameters in lower-precision formats. This has the effect of making the models even more lightweight at the cost of model accuracy. This tradeoff might be desirable, however, when running on limited hardware.



	LlaMA-3 8B	**Gemma 7B**
Layers	32	28
Model Dimension	4096	3072
FFN Dimension	14336	49152
Attention Heads	32	16
Key / Value Heads	8	16
Activation Function	SwiGLU	GeGLU
Vocabulary Size	128,000	256,128

Fig. 1. Model comparison between LLaMA-3 8B and Gemma 7B, adapted from [1] and [2].

A quick overview of each model is in order. LLaMA-3 is a LLM from Meta AI that natively supports multilinguality, coding, reasoning, and tool usage [1], while Gemma is an open-source LLM that supports question answering, common sense reasoning, STEM, and coding [2]. Both models’ architectures are based on a standard dense transformer; simply put, a transformer is a type of computer model that helps machines understand and generate human-like language by looking at parts of the input together, rather than just focusing on one part at a time.

The biggest differences between the models are within the activation function and vocabulary size. These activation functions mimic the neural processing of a human brain; SwiGLU utilizes a combination of non-monotonic function Swish and a “Gated Linear Unit” aka GLU [3], while GeGLU uses a GLU variant known as the “Gaussian Error Linear Unit” aka GELU for its neural network. The vocabulary size and feed-forward network (FFN) from Figure 1 are also worth noting. LLaMA-3’s model uses less dimensions as well as vocabulary range, limited to half the size of Gemma’s 256,128 words.

B. Research Objectives

The primary objective of this study is analyzing the performance of the LLaMA and Gemma models on the average user’s local hardware, from higher-end to lower-end computers. We would like to determine if comparable SLM performance can be seen in lower-end computers. This would both increase accessibility to SLMs as well as potentially increase the number of use cases for these lightweight models.

Due to the broad scope of the term performance, we break this down into two factors: technical and logical performance. The technical factor will assess overall latency and computational resource usage, while the logical will assess the accuracy and validity of the responses generated by each model.

We hypothesize that Gemma will outperform LLaMA in terms of accuracy in STEM-related questioning, but not in technical performance. On the other hand, LLaMA will have higher accuracy distributed across all fields as a jack-of-all-trades, not excelling in one particular topic. However, with its larger parameter size and number of layers, we expect LLaMA to be slower in latency.

II. METHODS

Two different machines were used to conduct this study, Daehee's laptop and Michael's desktop. Both machines use Windows. Daehee's laptop, which we will refer to as the higher-end machine, has the following specifications:

- NVIDIA GeForce RTX4080 (12GB) GPU
- Intel i9-13900HX (2.20GHz) CPU
- 16 GB RAM

Michael's desktop, which we will refer to as the lower-end machine, has the following specifications:

- NVIDIA GeForce GTX 1660 Ti (6 GB) GPU
- Intel i7-6800K (3.40 GHz) CPU
- 16 GB RAM

The two specific models of LLaMA and Gemma used in this study were *llama-3-8b-Instruct-bnb-4bit* and *gemma-7b-it-bnb-4bit*, respectively, and were retrieved from Hugging Face. Code was written and run using Python. Data was analyzed and graphed using Microsoft Excel.

A. Evaluating Accuracy

The logical performance of these models were evaluated using three standardized benchmarking tools: MMLU (Massive Multitask Language Understanding), GSM-8K (Grade School Math), and ARC (AI2 Reasoning Challenge). Each benchmark tests a different aspect of the model's capabilities, ranging from answering various high-school level topics like social sciences, humanities, mathematics, to multiple choice style questioning.

The MMLU benchmark tests the model's world knowledge and problem solving ability, spanning over 57 tasks [5]. These prompts consist of questions from freely available sources online, including practice questions ranging from the high school GRE to the U.S. medical licensing examinations. Questions are categorized by difficulty from "elementary", "high school", "college" and "professional" levels. Accuracy values were calculated by subject plus an overall value, based on the percentage of multiple-choice questions answered correctly. Constraints of 0-shot (no context), 5-shot (some context), and 10-shot (most context) were set to test accuracy levels according to the amount of hints and contextual information given.

The GSM-8K benchmark tests the model's capability to process grade-school worded math problems, requiring between 2 to 8 steps to solve each one [6]. Accuracy is measured by detection of the absolute correct value within a generated response (ex. 68+82+50=*200 gallons*). The default question count is set to 1319, but due to limitations in computational resources, the number of problems were configured to strictly 200 with a 5-shot context before each test was run.

Lastly, the ARC benchmark tests deeper text comprehension with fewer hints within the prompts; questions derive from a collection of 7787 standardized test questions within the natural science field [7]; structured in multiple choice format, this test measures accuracy by the percentage of correct choices given. Constraints of 0-shot (no context), 5-shot (some context), and 10-shot (most context) were set to test accuracy levels according to the amount of hints and contextual information given.

These benchmarks were conducted with assistance from EleutherAI's *lm-evaluation-harness* [4]. Certain constraints were put in place for consistent performance throughout each benchmark, such as limiting batch_size to 2 for all tasks, and the limit to 200 specifically for the GSM-8K. It should be noted that these benchmarks were only run on the higher-end machine. As these benchmarks only assess the accuracy of the models, comparison with the lower-end machine is irrelevant.

B. Evaluating Technical Performance

In order to evaluate technical performance, we gave the models two tasks: creative writing and summarization. We chose these tasks because their computational intensity would allow us to observe the models' performance under heavy load. For creative writing, the models were prompted to write three creative, short stories with specific premises. For summarization, the models were prompted to summarize Homer's *The Odyssey* and two other passages, one from a scientific journal article and one from the Wikipedia page on the theory of relativity. We attempted to choose a variety of topics in case the models were better suited to any specific subjects. For both tasks, the models were instructed to limit their responses to less than 2,000 characters.

Technical performance was assessed on both the higher-end and lower-end machines. We recorded five key technical metrics as the models performed the tasks: latency, CPU utilization, GPU utilization, GPU memory utilization, and GPU memory used. Latency, in this case, refers to the time it takes for the models to finish answering all prompts for a task and was measured in seconds. GPU utilization refers to how much of a GPU's processing power is currently being used, expressed as a percentage. GPU memory utilization refers to how often a GPU's memory is being read from or written to, expressed as a percentage. GPU memory used refers to the amount of memory occupied in VRAM, usually expressed in MB or GB.

To collect the technical performance data, we first had to install the libraries necessary for working with the SLMs. After that initial setup, we would simply run our Python script from the command line and select the model and task to perform. The model would then be loaded into memory to respond to the three prompts for the task. After the model finishes responding, the Python script closes. It is important that the script closes in order to free up memory for the next model and task. During that process, the script would periodically record a snapshot of the five technical metrics to .csv and .txt files.

After we gathered the raw data, we used Microsoft Excel to analyze and graph. We calculated means for the five technical metrics measured and calculated percentage differences between them. When calculating means, only data points where both GPU utilization and GPU memory utilization were nonzero values were included; these data points represent time when the models were active.

C. Code Design

Our Python script for evaluating the technical performance of the two SLMs can be divided into four main components: user input, data collection, model setup, and chatting with the model.

Because we were evaluating technical performance on two different machines and had different datasets for the two tasks, we knew that we were going to have a lot of different data files. At the end, we had a total of 16 files of data that needed to be analyzed. We wanted to have a way of easily identifying the source of each file. To achieve this, we had our script record user input of which model was being run, with which task, and on whose computer. The script would then determine the proper filename for when data was written to the .csv and .txt files. This made it a lot easier to analyze the data later, since for each file, we could immediately identify which model, task, and machine the data came from. Additionally, this made it exceptionally convenient to run all of the tests, since all we had to do was choose the model and task to run for each trial.

In order to collect data on the five technical metrics, we wrote the monitor_usage_to_csv function that utilizes the psutil and pynvml libraries. Psutil allows us to get the CPU utilization of our script, and pynvml, a library from NVIDIA, allows us to get the GPU metrics of our machines. Our function works by periodically taking snapshots (every three seconds) of the CPU and GPU metrics and writing those to a .csv file. It is important to note that this function has to be run in a new thread. Otherwise, the program will endlessly loop while writing the performance metrics to a .csv file. We create the thread and have the program sleep for 10 seconds to get a baseline measurement before setting up the model.

For setting up and chatting with the model, we would like to credit Jacob for his Python script for working with SLMs on local machines [8]. First, we set up the model by creating a pipeline and passing in arguments for model name, task, tensor data type, and a low CPU usage tag. Setting tensor data type to float 16 as well as using quantization makes the models more lightweight. With the pipeline created, we format our prompt with a chat template function and then pass it into the pipeline to start chatting with the model. To streamline data collection, we separate our six prompts into two lists by task and chat with the models by looping through the lists. Each time a model completes a prompt, the latency is recorded; all three latencies for the task are written to a .txt file once the model completes the task. During data analysis, we calculated the total latency for each task for each model and compared.

III. RESULTS

A. Evaluating Accuracy

Figure 2 represents the results of the MMLU tests; Gemma 7B reported at between 41.99% and 57.1% accuracy throughout with a 50.9 minute evaluation time, scoring highest in the other category. LLaMA-3 8B reported at between 52.01% and 71.86% accuracy throughout with a 32.8 minute evaluation time, scoring highest in the social sciences category. On average, the Gemma model scored a

49.46% accuracy rate compared to the outperforming LLaMA model at 61.46%.

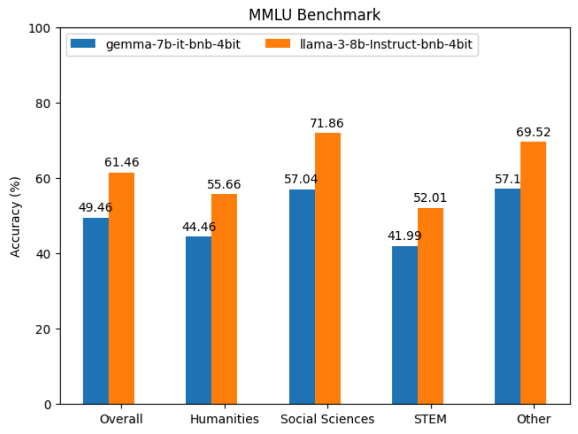


Fig. 2. MMLU benchmark results.

Figure 3 represents the results of the GSM-8K benchmark; with the 5-shot contextual constraint and limit of 200 questions, the Gemma model reported a 21.0% accuracy rate after 50.4 minutes, while the LLaMA model clocked a 69.5% accuracy after 21.6 minutes.

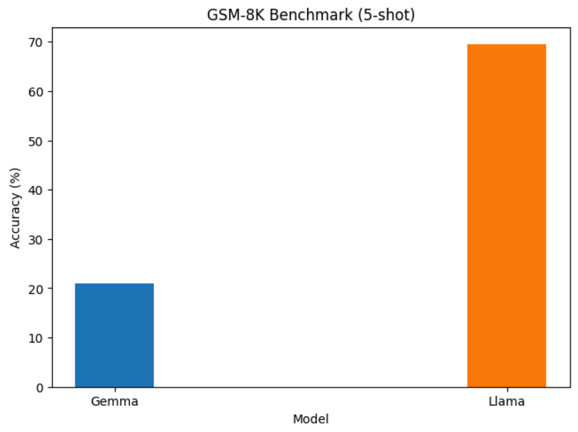


Fig. 3. GSM-8K benchmark results.

Figure 4 represents the last accuracy benchmark with the AI2 Reasoning Challenge. The Gemma model clocked in at 49.4% accuracy in 8.9 minutes for a 0-shot test, 50% accuracy in 11.5 minutes for a 5-shot test, and considerably lower 37.88% accuracy in 3.4 minutes for a 10-shot test. On the other hand, LLaMA-3 reported a 54.86% accuracy in 4.9 minutes for a 0-shot test, 59.56% accuracy in 9.7 minutes for a 5-shot test, and a 59.98% accuracy in 11.3 minutes for a 10-shot test.

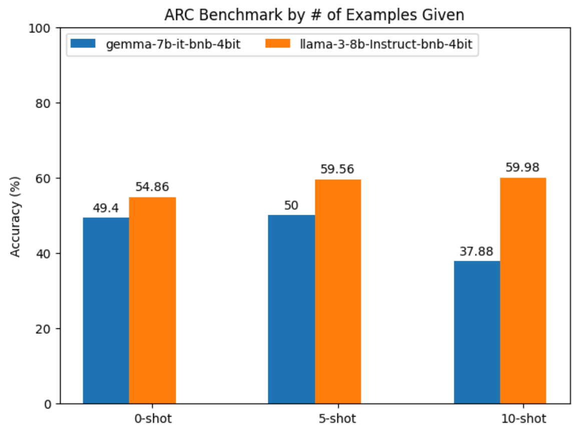


Fig. 4. ARC benchmark results.

B. Evaluating Technical Performance

According to Table 1, LLaMA performed significantly faster than Gemma on both the creative writing and summarization tasks on the lower-end machine. LLaMA performed 63.18% faster on the creative writing task and 41.02% faster on the summarization task compared to Gemma. Except for the summarization task, where LLaMA had 34.83% less mean GPU memory utilization, there were no major differences (% differences were all < 5%) in CPU or GPU technical performance between Gemma and LLaMA on the lower-end machine.

TABLE I
LOWER-END MACHINE TECHNICAL PERFORMANCE

	Creative Writing			Summarization		
	Gemma	Llama	% Difference	Gemma	Llama	% Difference
Latency (s)	1,023	631.2	63.18	387.7	275.0	41.02
CPU Util. (%)	11.69	11.98	2.433	11.68	11.42	2.282
GPU Util. (%)	98.18	95.56	2.752	93.63	95.73	2.241
GPU Mem. Util. (%)	5.916	6.110	3.273	6.628	4.916	34.83
GPU Mem. Used (MB)	6,073	6,016	0.9398	5,953	5,976	0.3933

According to Table 2, LLaMA performed significantly faster than Gemma on the summarization task with a 34.46% difference in speed on the higher-end machine. However, there was no major difference in latency for the creative writing task. Additionally, there were no major differences in either mean CPU utilization or mean GPU memory used. On the other hand, mean GPU utilization and mean GPU memory utilization significantly differed in both tasks for the higher-end machine. For creative writing, LLaMA had 38.31% less GPU utilization and 39.15% less GPU memory utilization. For summarization, LLaMA had 32.38% less GPU utilization and 38.17% less GPU memory utilization.

TABLE II
HIGHER-END MACHINE TECHNICAL PERFORMANCE

	Creative Writing			Summarization		
	Gemma	Llama	% Difference	Gemma	Llama	% Difference
Latency (s)	57.17	59.22	3.586	21.11	15.70	34.46
CPU Util. (%)	2.802	2.749	1.931	2.979	2.946	1.097
GPU Util. (%)	64.11	46.35	38.31	68.57	51.80	32.38
GPU Mem. Util. (%)	46.89	33.70	39.15	50.57	36.60	38.17
GPU Mem. Used (MB)	6,600	6,305	4.685	6,543	6,301	3.840

We also compared the technical performances of the higher-end machine to the lower-end machine by looking at how Gemma performed on the creative writing task on each machine. In terms of latency, the lower-end machine was 1,689% slower than the higher-end machine and finished the task in 1,023 seconds compared to 57.17 seconds. Looking at mean CPU utilization, Figure 5 shows how the lower-end machine had significantly higher (317.2%) mean CPU utilization compared to the higher-end machine.

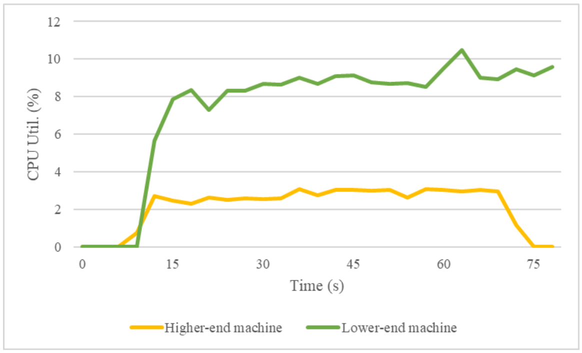


Fig. 5. Graph comparing CPU utilization using Gemma and creative writing. (mean % difference = 317.2%).

Mean GPU memory utilization also differed significantly between the two machines. Figure 6 shows how the higher-end machine had significantly higher (692.6%) mean GPU memory utilization compared to the lower-end machine throughout the duration of the task. Note that the figure does not graph the entire duration of the task for the lower-end machine; the purpose of the figure is to illustrate the trend in GPU memory utilization for the lower-end machine.

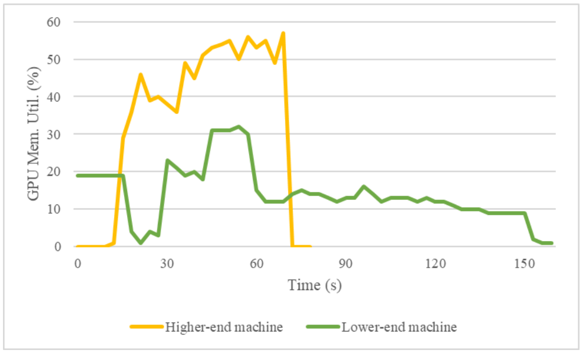


Fig. 6. Graph comparing GPU memory utilization using Gemma and creative writing. (mean % difference = 692.6%).

Finally, mean GPU utilization differed significantly between the two machines. Figure 7 shows how for the lower-end machine, mean GPU utilization stays high near 100% for the entirety of the task's duration. This represents 53.14% higher mean GPU utilization for the lower-end machine compared to the higher-end machine.

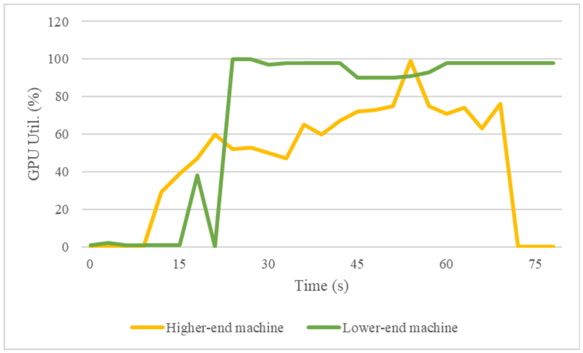


Fig. 7. Graph comparing GPU utilization using Gemma and creative writing. (mean % difference = 53.14%).

IV. DISCUSSION

A. Evaluating Accuracy

Following each benchmark test, the Gemma model surprisingly reported considerably lower accuracy despite its higher vocabulary size and FFN dimensions. An interesting result to consider is Gemma's accuracy rates in the ARC results; at 0 and 5-shot contexts, Gemma performed scored 49.4%, which negligibly increased to 50%, respectively. Once the context rose to 10-shot, the accuracy rate of the Gemma model dropped sharply to 37.88%; a plausible cause is that greater context conflicted with the information that the Gemma model was trained with, causing higher rates of what is known as "hallucinations" (a phenomenon where a language model generates false or misleading information that is either not based on training data or an incorrect decoding from the transformer). Nevertheless, the findings from Gemma's results did not support our hypothesis that it would outperform LLaMA in STEM-related questions.

In regards to LLaMA's results, the model performed consistently well in all benchmark tests, averaging an overall 61.6% accuracy rate; this can be attributed to less garbage parameters and better training. Logical performance showed an even distribution throughout each benchmark, with an exception within MMLU's results in the social sciences and "other" category in Figure 2. This can be attributed to the fact that the "other" category consisted of niche information that could not be categorized as easily, such as U.S. Medical License examination and miscellaneous statistics. These findings support our hypothesis of LLaMA-3 being a jack of all trades.

B. Evaluating Technical Performance

The results do not support our hypothesis that LLaMA would be slower in latency. In fact, LLaMA performed faster than Gemma in three out of the four datasets on both machines with the fourth result being a tie with no major difference.

An interesting result is that even though there were no major differences in CPU or GPU metrics on the lower-end machine, except for the summarization task where LLaMA actually had 34.83% less mean GPU memory utilization, LLaMA still performed significantly faster. This result may highlight the superiority of LLaMA's architecture. Recommended system requirements for both SLMs include a minimum of 6 GB of VRAM. Because the lower-end machine barely has enough VRAM, the SLMs have to offload some processing to the lower-end machine's CPU. The issue is that CPUs are not well-suited for simultaneous processing, which SLMs require. The result is a drastic decrease in performance. However, even with this drastically reduced performance, LLaMA still significantly outperformed Gemma in terms of latency, highlighting the differences in their architecture.

The results from the higher-end machine further illustrate these differences. Although LLaMA only performed significantly faster on the summarization task, it had significantly less mean GPU utilization and mean GPU memory utilization for both tasks. This means that with fewer resources, LLaMA was able to outperform or at least match Gemma. These results both support our hypothesis that LLaMA would outperform Gemma in technical performance.

As mentioned, the lower-end machine lacks sufficient VRAM to hold the entirety of the SLMs, leading to GPU offloading to the CPU and thus, higher mean CPU utilization. This explains the difference in mean CPU utilization between the two machines that we see in Figure 5. Because the higher-end machine has sufficient VRAM and does not need to GPU offload, the GPU handles most, if not all, of the processing, thus leading to lower mean CPU utilization. As mentioned previously, CPUs are not well-suited for the simultaneous processing that SLMs require. Thus, we believe that this GPU offloading and lack of sufficient VRAM are the main reasons that the lower-end machine's performance was so much slower (1,689% slower) compared to the higher-end machine.

Another issue that can cause GPU offloading is maximum GPU utilization. We can see in Figure 7 that although the higher-end machine has a high mean GPU utilization, it does not sustain maximum GPU utilization for long, if at all. On the other hand, the lower-end machine is almost always at maximum GPU utilization. The issue is that if all of the GPU's processing power is occupied, then the SLM must rely on the CPU for additional processing, which is not ideal.

Lastly, if we look at Figure 6, we see that the lower-end machine has a lower mean GPU memory utilization. Recall that GPU memory utilization refers to how often a GPU's memory is being read from or written to. Even though the lower-end machine is almost constantly at maximum GPU utilization, its mean GPU memory utilization is quite low over the duration of the task. We suspect that at the beginning of the task, the GPU's processing power is not yet fully occupied and it processes as normal. As time goes on, the GPU has to eventually offload to the CPU, which causes a bottleneck. The GPU is then likely waiting on data being processed by the CPU, thus causing the decrease in mean GPU memory utilization. On the other hand, the higher-end machine does not have to deal with this bottleneck, and we can see that it has a high, sustained GPU memory utilization rate.

V. CONCLUSION

In conclusion, the Gemma and LLaMA-3 models had starkly different performance results. Gemma's model performed much less favorably in both technical and logical performance, while LLaMA performed at faster speeds and with greater accuracy. A valid shortcoming to consider is that the particular model of LLaMA is Meta's 3rd iteration, compared to Gemma with its first iteration. With this in mind, a follow-up study assessing LLaMA-3 with Gemma's 3rd iteration could show a much more competitive side-by-side comparison in use cases for lower-end machines. Additionally, a future study should be conducted comparing the performance between an even higher-end machine and a machine that has just enough specifications to run the SLMs we studied without GPU offloading. This would truly illustrate how much of a difference hardware has on SLM performance and if more accessible, low-cost setups are comparable.

GROUP MEMBERS' CONTRIBUTIONS

- Planning and design
  - Michael, Daehee
- Code writing
  - Michael, Daehee
- Accuracy evaluation
  - Daehee
- Technical performance evaluation
  - Michael
- Data analysis
  - Michael, Daehee
- Report write-up

        - Michael, Daehee
- Demo videos
    - Michael, Daehee
- Presentation and slides
    - Michael, Daehee

REFERENCES

1. Meta, “Introducing Meta Llama 3: The most capable openly available LLM to date,” *ai.meta.com*, Apr. 18, 2024. https://ai.meta.com/blog/meta-llama-3/ (accessed Aug. 11, 2025).
2. Gemma Team *et al.*, “Gemma: Open Models Based on Gemini Research and Technology,” *arXiv.org*, Mar. 13, 2024. https://arxiv.org/abs/2403.08295 (accessed Aug. 11, 2025).
3. Selssabil, “Exploring SwiGLU : The Activation Function Powering Modern LLMs,” *Medium*, Oct. 04, 2024. https://medium.com/@s_boudefel/exploring-swiglu-the-activation-function-powering-modern-llms-9697f88221e7 (accessed Aug. 11, 2025).
4. EleutherAI, “EleutherAI/lm-evaluation-harness,” *GitHub*, Mar. 18, 2024. https://github.com/EleutherAI/lm-evaluation-harness (accessed Aug. 11, 2025).
5. D. Hendrycks et al., “Measuring Massive Multitask Language Understanding,” Jan. 2021. Accessed: Aug. 11, 2025. [Online]. Available: https://arxiv.org/pdf/2009.03300
6. K. Cobbe *et al.*, “Training Verifiers to Solve Math Word Problems,” *arXiv (Cornell University)*, Oct. 2021, doi: https://doi.org/10.48550/arxiv.2110.14168.
7. P. Clark et al., “Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge,” Mar. 2018. Accessed: Aug. 11, 2025. [Online]. Available: https://arxiv.org/pdf/1803.05457
8. J. Jacob, “Assessing the SLM versions of LLAMA 3, Gemma, and Mistral for local laptop usage,” Medium, May 11, 2024. https://medium.com/@jaimonjk/assessing-the-slm-versions-of-llama-3-gemma-and-mistral-for-local-laptop-usage-58311ce189f4 (accessed Aug. 11, 2025).