LLM Performance Optimized by DeepSpeed

Disha Agarwal
dagarw24@asu.edu
Arizona State University
Tempe, Arizona, USA

Uma Maheshwara Swamy Desineedi
udesinee@asu.edu
Arizona State University
Tempe, Arizona, USA

Kathan Shah
kshah78@asu.edu
Arizona State University
Tempe, Arizona, USA

ABSTRACT

This paper presents a project proposal for CSE 598: Data Intensive Systems for Machine Learning (DISML), focusing on the performance evaluation of fine-tuning Large Language Models (LLMs) with DeepSpeed. The study aims to address the gap in research regarding the impact of LLM model architecture (encoder-only, decoder-only, or transformer models) on system enhancements facilitated by an optimizer. Specifically, we investigate how different LLM architectures interact with DeepSpeed, a framework designed to optimize storage, compute, and other resources. The experimentation involves baseline fine-tuning and inference tasks without using an optimizer, comparing them to the improvements in resource utilization with DeepSpeed. Two LLM models, including a decoder-only (GPT-like) model and a transformer model (comprising both encoder and decoder), are evaluated using SciTail[6] and Multi-XScience[7] datasets. The proposed evaluation metrics include the time taken when running inference on these models.

KEYWORDS

Deepspeed, LLM.etc.

1 INTRODUCTION

Large Language Models (LLMs) have revolutionized natural language processing (NLP) tasks, showcasing remarkable capabilities in understanding, generating, and processing textual data. However, the training and fine-tuning of these models present significant computational challenges, demanding extensive computational resources and time. To address these challenges, specialized frameworks such as DeepSpeed have emerged, aiming to optimize efficiency across various dimensions including compute, memory, and communication.

Despite the advancements in optimizing LLM training pipelines, there remains a gap in understanding how different LLM architectures interact with optimization frameworks like DeepSpeed. Specifically, there is limited research investigating the impact of model architecture—such as encoder-only, decoder-only, or transformer models—on the effectiveness of optimizers. This gap in knowledge hinders the establishment of best practices for utilizing LLMs efficiently, particularly given their widespread adoption and usage across various domains.

This research aims to bridge this gap by conducting a comprehensive study on the performance evaluation of fine-tuning large language models with DeepSpeed. The focus will be on investigating how different LLM architectures, namely decoder-only and transformer models, interact with DeepSpeed to maximize resource utilization and training efficiency.

The proposed research will contribute to advancing our understanding of the interplay between LLM architectures and optimization frameworks, thereby informing best practices for efficient training and fine-tuning of large language models. By evaluating performance metrics such as memory usage, fine-tuning time, and throughput, we aim to provide actionable insights that can guide practitioners and researchers in optimizing LLM workflows for enhanced efficiency and scalability.

This report has the following format. The section on related work gives a brief intro to deepspeed and LLM optimization using deepseed. The dataset section describes the datasets and tasks included for the experimentation of our project. The methodology mentions the experiment setup and challenges faced. The result of the model inference is in the result section. The future work section details the scope of future research and possible extension of this project.

2 RELATED WORKS

As deep learning models expand in size, they offer significant improvements in accuracy [9]. In the realm of natural language processing (NLP), the advent of transformers has led to the development of large-scale models such as Bert-large (0.3B), GPT-2 (1.5B), Megatron-LM (8.3B), and T5 (11B). However, as we aim to

push the boundaries further and scale these models to trillions of parameters, we encounter challenges in training or fine-tuning them [10]. These challenges include the limitation of fitting such large models into the memory of a single device, such as a GPU or TPU. Simply adding more devices does not suffice to scale the training process effectively.

2.1 Data, Model and Pipeline Parallelism

Conventional data parallelism (DP) fails to reduce memory per device and reaches memory limitations for models exceeding 1.4B parameters on GPUs with 32 GB memory. Alternative approaches like Pipeline Parallelism (PP), Model Parallelism (MP), CPU-Offloading, etc., involve trade-offs between functionality, usability, memory, and compute/communication efficiency. Balancing these factors is essential for achieving both speed and scalability in training.

Among the various existing solutions for training large models, Model Parallelism (MP) stands out as particularly promising. However, its scalability is limited beyond certain model sizes. MP divides the model vertically, distributing computation and parameters across multiple devices within each layer, necessitating substantial communication between layers. Consequently, its efficiency declines rapidly beyond a single node. Research indicates that a 40B parameter model utilizing Megatron-LM across two DGX-2 nodes achieves approximately 5 T flops per V100 GPU, which is less than 5% of hardware peak performance [11].

To address the constraints of current solutions and enhance the efficiency of training large models, a full-spectrum of memory consumption on model training is studied and classified into two main categories: 1) the majority of memory is occupied by model states, encompassing optimizer states, gradients, and parameters, and 2) the remaining memory is utilized by activations, temporary buffers, and fragmented memory, collectively termed residual states. Advanced techniques have been introduced to mitigate these limitations, with our study focusing on ZeRO (Zero Redundancy Optimizer) to optimize memory efficiency while maintaining high compute and communication efficiency. Subsequently, we investigated the impact of ZeRO on the performance of different model architectures.

2.2 DeepSpeed

These methods all maintain the complete set of model states needed throughout the entire training process, even though not all states are always required during training. Building on these insights, ZeRO-DP (ZeRO-powered data parallelism) [3] has been developed to combine the computational and communication efficiency of data parallelism (DP) with the memory efficiency of model parallelism (MP). ZeRO-DP achieves this by eliminating redundant memory states across data-parallel processes through partitioning rather than replication. It preserves computational and communication efficiency by maintaining the computational granularity and communication volume of DP using a dynamic communication schedule during training. ZeRO-DP consists of three primary optimization stages: 1) Optimizer State Partitioning, 2) Add Gradient Partitioning, and 3) Add Parameter Partitioning. When all three stages are enabled, ZeRO can effectively train a trillion-parameter model. ZeRO is further optimized for increased CPU and GPU utilization with ZeRO-Offload and low bandwidth, small batch-size per GPU with ZeRO++ [4] to overcome the challenges of ZeRO technique in specialized environments. We utilize various techniques of ZeRO for our evaluations using the metrics as defined in [2].

3 DATA

3.1 Dataset Description

We use 2 datasets which are a subset of the BigBIO framework [5] consisting 2 tasks NLI and Summarization

3.1.1 Scitail[6]: The Allen Institute for AI created SciTail, which is made up of pairs of entailments that are drawn from questions and replies pertaining to science. The dataset, which includes 27,026 question-answer pairs divided into entailment and non-entailment groups, can be downloaded from the Hugging Face website. It is an indispensable tool for teaching Machine Learning models to recognize entailment relationships, particularly in scientific settings. We have also given an analysis of the input token length to understand how long a model may take to process it.

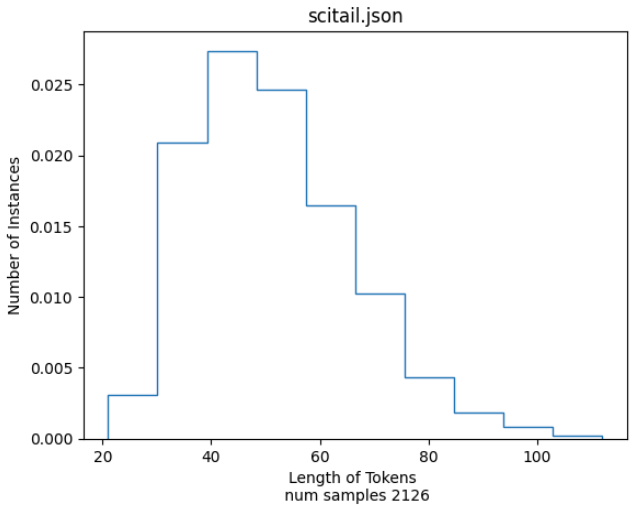

Figure 1: Dataset token distribution for scitail

3.1.2 Multixscience[7]: The summary dataset Multi-XScience poses a difficult task: given an abstract, produce related work. This dataset, which has over 30,000 training samples, is available on the Huggingface website. We plot the token length for input to understand the nature of the task.

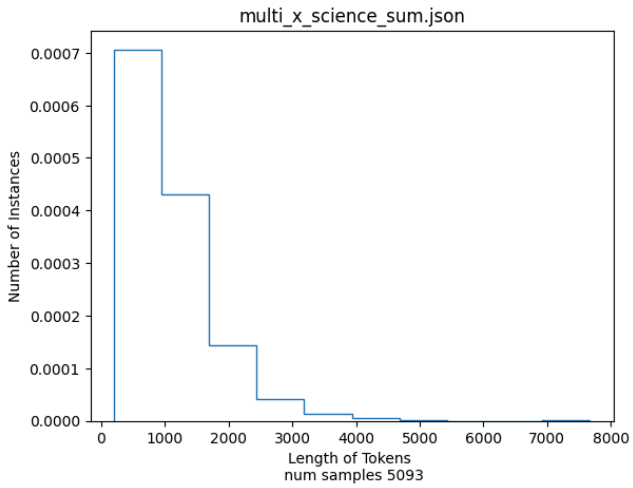

Figure 2: Dataset token distribution for multi_x_science

4 METHODOLOGY

LLMs are widely used for NLP tasks for text generation, summarization, and classification. LLms consume a lot of computing and memory resources and need to be optimized for environmental and efficiency purposes. Here we evaluate the time needed to run inference on LLMs in Google Colab specifically for the biomedical domain.

4.1 Data Processing and Exploratory Data Analysis (EDA)

We explored the data using exploratory data analysis to better understand it. All datasets were first normalized, with input attributes and output labels combined into two columns called input and output. After that, we looked at the value counts in each column to identify distributions and classifications. Also, we performed a token length analysis to understand the tasks and their complexity.

4.2 Flan-T5 Model

FLAN-T5[10] is an encoder-decoder model fine-tuned for text-to-text tasks, leveraging instruction prompts for its training regimen. Developed and refined for over 1.8 tasks, FLAN-T5 exhibits remarkable versatility in handling various natural language processing tasks, including summarization and classification. Its architecture is based on the T5 model, renowned for its efficacy in generating high-quality outputs across a wide range of NLP tasks. FLAN-T5's fine-tuning process emphasizes adaptability to diverse tasks and data, ensuring robust performance in real-world applications. With its ability to effectively process and generate text based on instruction prompts, FLAN-T5 represents a significant advancement in the field of natural language processing, offering promising opportunities for tackling complex language understanding and generation tasks.

Flan-t5 has different versions from small, base to large, and xl size models. For this project, we use the small and base models with sizes of 60 million and 700 million respectively. We used the model in generate mode, with no classification head for both tasks.

Deepspeed Optimization: We used zero optimization configuration with the optimization stage as stage 3. Within this stage, key optimizations include offloading the optimizer and parameter states to the CPU, enabling memory pinning for efficient data transfers. Additionally, communication overlap and gradient contiguity optimizations are enabled to enhance training efficiency.

```
{
    "zero_optimization": {
      "stage": 3,
      "offload_optimizer": {
          "device": "cpu",
          "pin_memory": true
      },
      "offload_param": {
          "device": "cpu",
          "pin_memory": true
      },
      "overlap_comm": true,
      "contiguous_gradients": true,
      "sub_group_size": 1e9,
      "reduce_bucket_size": "auto",
      "stage3_prefetch_bucket_size": "auto",
      "stage3_param_persistence_threshold": "auto",
      "stage3_max_live_parameters": 1e9,
      "stage3_max_reuse_distance": 1e9,
      "stage3_gather_fp16_weights_on_model_save": true

    },
    "train_batch_size": 4
}
```

Figure 3: Deepspeed Configuration

4.3 Llama Model

Llama 2[9] contains several Large Language Models (LLMs) that have been trained and optimized, ranging in parameter values from 7 billion to 70 billion. Text summarization is one of the many content kinds that these models can produce. The foundational architecture of Llama 2, created by Meta

AI, is a transformer model that has been improved via fine-tuning and reinforcement learning with human feedback (RLHF). It is a useful tool for people and organizations who handle large amounts of textual data. Human assessments of Llama 2's helpfulness and safety show that it performs better, is more efficient and is more versatile than closed-source models like Falcon, MBT, ChatGPT, and Bard. Llama 2 was trained on extensive text datasets, which included 2 trillion tokens and over 1 million new human annotations.

We use the 7b version quantized to 8-bit representation. Since, Llama is a generative model, for the NLI task we added “output only the label” to the prompt. For Multi-x-science the prompt was to generate the related work given the abstract of the current paper and the cited paper.

5 RESULTS & CONCLUSION

We experimented with 2 types of models and 2 types of tasks. The 2 tasks were NLI and text generation. As expected NLI took lesser time than text generation. *For all the experiments Google Colab single free GPU environment was used*.

Flan-T5 Model: We experimented with the small and base model with and without deepspeed optimization configuration in the Google colab environment with a single GPU. For, these models we observed that deepspeed optimization did not cause any improvement in the time used for inference. This may be because we used smaller model sizes hence the observed difference was not significant. Furthermore, we experimented with inference and not fine-tuning and the configuration might be more suited for fine-tuning and not inference. The results are listed in Table 1.

LLama Model: We experimented with an 8-bit quantized model in colab, we tried to use it for deepspeed optimization but we discovered that since the model had already been quantized deepspeed configuration couldn't optimize it further.

Table 1: Results of inference on Google Colab

Model	**Dataset**	**Deepspeed yes/ no**	**Time**
google/flan-t5-small	Scitail	no	3.9ms
google/flan-t5-small	Multi-x-science	no	44.7ms
google/flan-t5-small	Scitail	yes	9.6ms
google/flan-t5-small	Multi-x-science	yes	113.8ms
google/flan-t5-large	Scitail	no	29.2ms
google/flan-t5-large	Multi-x-science	no	389.7ms
google/flan-t5-large	Scitail	yes	29.5ms
google/flan-t5-large	Multi-x-science	yes	494.8ms

Model	DS ZeRO offload	Hardware	batch size per GPU	precision	duration	cost
FLAN-T5-XL (3B)	No	4x V100 16GB	OOM	fp32	-	-
FLAN-T5-XL (3B)	No	8x V100 16GB	1	fp32	105h	~$2570
FLAN-T5-XL (3B)	No	8x A100 40GB	72	bf16	2,5h	~$81
FLAN-T5-XL (3B)	Yes	4x V100 16GB	8	fp32	69h	~$828
FLAN-T5-XL (3B)	Yes	8x V100 16GB	8	fp32	32h	~$768
FLAN-T5-XXL (11B)	Yes	4x V100 16GB	OOM	fp32	-	-

Figure 4: Results for larger models courtesy:https://www.philschmid.de/fine-tune-flan-t5-deepspeed.

As we can see in Figure 4, we expect to get similar results on our datasets if we run inference and fine-tuning experiments on larger flan-t5 models and multi-GPU environments which are expensive to set up.

Also, we tried using SOL but ran into errors with the installation of libraries with the Python environment(we did not spend much time on debugging this as the setup was working in colab) and hence stuck to colab.

To sum up, key findings have been that deepspeed zero configuration doesn't show time improvement for smaller

models on inference of both tasks. At least for the Google colab setup we used.

6 FUTURE WORKS

The present study initiates a foundational exploration into the intricate synergy between Large Language Model (LLM) architectures and the DeepSpeed framework, with a primary focus on performance optimization. Delving beyond the purview of ZeRO optimization techniques, DeepSpeed offers a wealth of additional functionalities ripe for exploration. Future research endeavors can delve into the nuanced impact of features such as gradient accumulation, mixed-precision training, and activation checkpointing on the performance of LLM architectures within the DeepSpeed framework.

While our study's lens primarily concentrates on evaluating inference time, future research horizons beckon towards broader applications in real-world Natural Language Processing (NLP) tasks. Tasks like question answering, summarization, or machine translation present fertile ground for benchmarking DeepSpeed-optimized LLM architectures. By extending evaluations to encompass these practical applications, researchers can gain invaluable insights into how optimization techniques translate into tangible improvements in application-specific performance metrics.

As the boundaries of deep learning continue to expand, scalability emerges as a pivotal consideration. Future research can explore the scalability of DeepSpeed-optimized LLM architectures to larger model sizes, potentially extending into the realm of models housing trillions of parameters. Additionally, investigations into DeepSpeed's performance across advanced hardware platforms like TPUs or custom AI chips hold promise for uncovering insights into its adaptability across diverse computing environments.

Moreover, amidst the rich tapestry of optimization frameworks tailored for large-scale language models, comparative studies offer a compelling avenue for future exploration. By comparing DeepSpeed against alternative frameworks such as Megatron-LM or TensorFlow, researchers can illuminate the relative strengths and weaknesses of different optimization paradigms. Such analyses promise to inform practitioners on the optimal selection of frameworks for specific LLM architectures, fostering advancements in model efficiency and performance.

ACKNOWLEDGMENTS

We would like to express our gratitude to Dr. Jia Zou and the teaching assistant Soham Nag for providing us with this wonderful opportunity to work on this project.

REFERENCES

[1] BetterProgramming. 2022. Frameworks for Serving LLMs. BetterProgramming. https://betterprogramming.pub/frameworks-for-serving-llms-60b7f7b23407

[2] Databricks. 2022. LLM Inference Performance Engineering Best Practices. Databricks Blog. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices

[3] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, & Yuxiong He. (2020). ZeRO: Memory Optimizations Toward Training Trillion Parameter Models.

[4] Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Connor Holmes, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. 2023. ZeRO++: Extremely Efficient Collective Communication for Giant Model Training. ACM Trans. Comput. Syst. 42, 1, Article 1 (January 2023), 30 pages. DOI:https://doi.org/10.1145/1234567.1234567

[5] Fries J, Weber L, Seelam N, Altay G, Datta D, Garda S, Kang S, Su R, Kusa W, Cahyawijaya S, Barth F. Bigbio: a framework for data-centric biomedical natural language processing. Advances in Neural Information Processing Systems. 2022 Dec 6;35:25792-806.

[6] Pietro Lesci. Year. SciTail. Hugging Face Datasets. https://huggingface.co/datasets/pietrolesci/scitail

[7] Multi-X Science Sum. Hugging Face Datasets. https://huggingface.co/datasets/multi_x_science_sum?row=12

[8] Microsoft. 2024. DeepSpeed. GitHub Repository. https://github.com/microsoft/DeepSpeed

[9] Hugo Touvron, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (July 18, 2023)

[10] Hyung Won Chung, Le Hou,et al. 2022. Scaling Instruction-Finetuned Language Models. arXiv. DOI:10.48550/ARXIV.2210.11416

[11] Connor Holmes, Masahiro Tanaka et al. 2024. DeepSpeed-FastGen: High-throughput Text Generation for LLMs via MII and DeepSpeed-Inference. arXiv:2401.08671v1 [cs.PF]. January 9, 2024.