DELL Technologies

Whitepaper

Fine-tuning Language Models on Dell PowerEdge XE9680 Servers

Authors:
Ben Fauber, Ph.D.,
Senior AI Research Scientist
and Distinguished Technical Staff

Bhavesh Patel,
Senior Distinguished Engineer

Introduction to Large Language Models

Large language models (LLMs) are artificial intelligence (AI) systems that use machine learning (ML) algorithms to process vast amounts of natural language text data. They have become increasingly popular due to their impressive natural language processing (NLP) capabilities.[1] Large pretrained language models are capable of extracting generalizations from vast amounts of text data, which can be utilized for a myriad of downstream applications such as text classification, text summarization, text generation, named entity recognition (NER), text sentiment analysis, and question-answering (Q&A). Additionally, many large language models are multilingual, making them even more versatile in utilizing text datasets across many different languages.

One of the strengths of large language models is that they contain a broad amount of information and knowledge, thanks to the massive amount of text data used to train them. However, this also means that they often struggle as specialists for deeper dives on topics or items with limited instances in the training dataset.[2] To address this shortcoming, businesses can add domain-specific information from their vertical into a pre-existing large language model. This layered training approach in which specialized information is added to a pretrained model is known as transfer learning, or model fine-tuning.

Language Model Fine-Tuning

Transfer learning creates application-specific parameters on top of pretrained large language models. The process involves exposing a pre-existing model to new information, allowing the model to adapt to that new information while not forgetting the old information.[3] A neural network's weights are updated during transfer learning training, and the new weights may not be compatible with the old weights of the pretrained model. Thus, if transfer learning is not carefully executed, the model's ability to perform the original task may degrade significantly resulting in an outcome known as "catastrophic forgetting."[4]

Traditional approaches to transfer learning involved freezing all but a few layers of the deep neural network (DNN) of the pretrained model, thereby allowing only a few layers of the pretrained neural network to learn from the new data and avoid catastrophic forgetting.[5] This approach has been particularly effective in transfer learning with large transformer-based language models like BERT.[6]

ChatGPT has gained considerable attention since its initial release for its ability to write fluid human-like prose.[7] ChatGPT is a refinement of the InstructGPT large language model. InstructGPT was created through the alignment of large language model outputs with user intent by incorporating reinforcement learning from human feedback (RLHF).[8]

Application of reinforcement learning from human feedback to create InstructGPT involved three major steps: 1) fine-tuning the GPT-3 large language model on diverse user prompts and appropriate responses, 2) rewarding the fine-tuned model when it combined appropriate prompt and response pairs, and 3) scoring the alignment of random prompts and responses with user intent by applying a reinforcement learning (RL) model based on the outcomes of step 2. Ultimately, the performance of InstructGPT was compared against its predecessor, GPT-3, based on their ability to infer and follow user instructions (helpfulness), their tendency for hallucinations (truthfulness), and their ability to avoid inappropriate and derogatory content (harmlessness). InstructGPT outperformed GPT-3 on all three criteria, with human referees favoring the outputs of InstructGPT 85% of the time.

Memory usage and communication costs are important considerations when training large language models. Recent work has shown that selectively updating only a small subset of a model's parameters during transfer learning training can alleviate these issues while also avoiding catastrophic forgetting.[9] So-called parameter-efficient fine tuning (PEFT) has been deployed to rapidly fine-tune large language models with domain-specific data and up-to-date information.

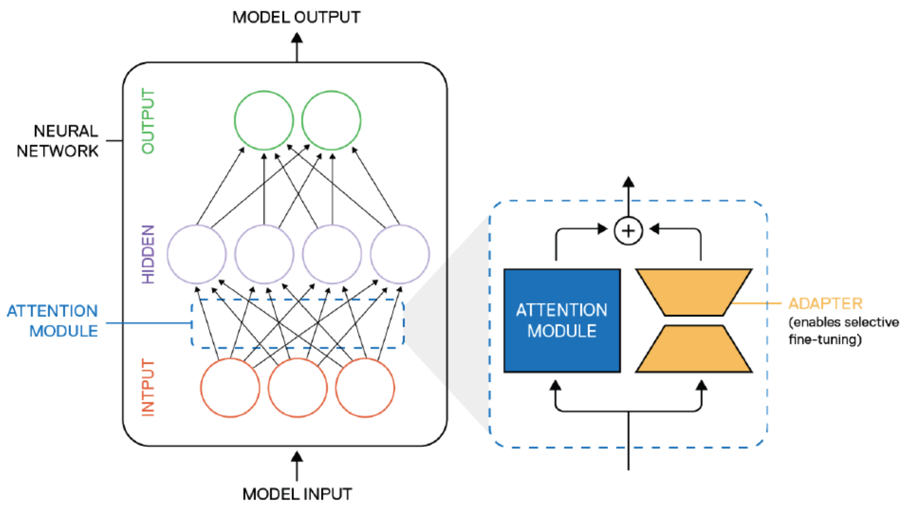


Figure 1. Illustration of the parameter efficient fine-tuning adapter technique applied to the attention module of a simplified shallow neural network.

Parameter-efficient fine-tuning leverages a regularization technique known as an adapter-based approach. The adapter technique inserts small bottleneck layers within each layer of a pretrained neural network model, thereby fixing the pretrained layers, and training the adapter layers on the new data (Figure 1).[10] This approach has been shown to improve model stability and robustness in transfer learning for various applications with minimal computational overhead. This method is particularly useful for fine-tuning large language models.

High-Performance Computing Infrastructure is Essential

Building large language models specific to a certain business or vertical with transfer learning requires in-house machine learning expertise or an engagement with a trusted partner. Additionally, high-performance computing (HPC) hardware infrastructure such as multiple graphics processing units (GPUs), high-speed networking, and structured datasets for transfer learning are required. A high-level understanding of the datasets used to train the existing large language models is essential to ensure that the new data used in transfer learning will impart information diversity and increased reach of the business-specific large language model. There are a few publicly available data sets such as The Pile[11] and ROOTS[12] for training large language models, yet many of the data sets used for training are proprietary and not public.

Fine-Tuning Language Models within Your Secure Infrastructure

There are several providers of large language model application programming interfaces (API) that allow businesses to programmatically connect directly to the model.[7] However, these application programming interfaces could have limited to no flexibility for a user to customize the language model to a specific business need or ontology. Further, the costs of operating compute-intensive large language models via third-party providers, instead of operating the same models on-premises, should be carefully considered, as routine on-prem operations can be more cost effective.

Enable your organization with large language models secured within your infrastructure to securely interact with your data and employees. Open-source foundational large language models such as BLOOM 176B,[13] BLOOMZ-176B,[14] GPT-J-6B,[15] or OPT-175B[16] can be instantiated within your corporate information technology (IT) infrastructure. The biggest benefit of running the model within your infrastructure is security: securely manage the model and secure the information it receives. Additional benefits include faster inference, greater availability and up time, reduced reliance on third-party services, maintain greater control of your own data and infrastructure, cost-effectiveness, and filtering the language model outputs to align them with your organization's domain and voice.

Empower your organization with large language models fine-tuned on your data and secured within your infrastructure. Copies of the large language model instantiated within your corporate IT infrastructure can be securely fine-tuned with domain-specific knowledge, continually updated with new data, and managed with guardrails on content, bias, and toxicity to create your organization's customized large language models. Dell Technologies can help you on this journey.

Recent Advances in Fine-Tuning Language Models

Recent results from a research group at Stanford University demonstrated the benefits of using a large language model to further fine-tune and refine an existing large language model, resulting in a new model known as ALPACA.[17] This process demonstrated a new paradigm in which one large language model was used to train another large language model. This approach could further reduce the role of humans in the model training loop, such as reinforcement learning with human feedback (RLHF), as previously leveraged by InstructGPT and ChatGPT.[8]

In ALPACA's case, a large generalist language model created a curated series of questions and answers on which to fine-tune another. Fine-tuning of the second language model with questions generated by the first resulted in improved performance metrics of the second language model. This paradigm has since been expanded to other domains outside of question and answer to enable rapid and efficient tuning of existing large language models on new data.[18]

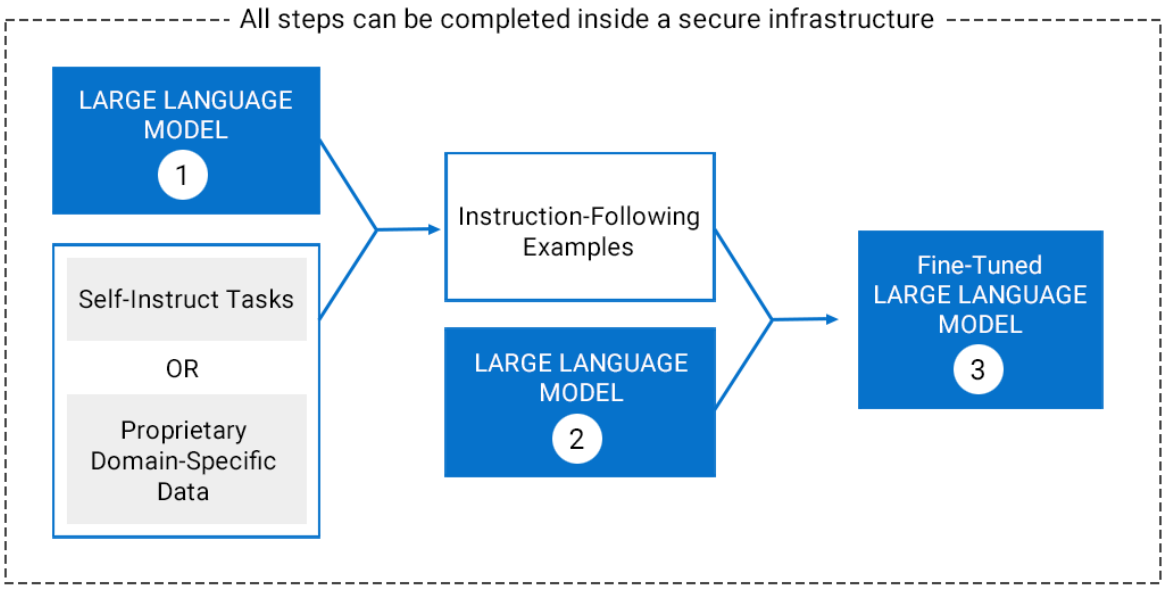

All steps can be completed inside a secure infrastructure
LARGE LANGUAGE MODEL 1
Self-Instruct Tasks
OR
Proprietary Domain-Specific Data
Instruction-Following Examples
LARGE LANGUAGE MODEL 2
Fine-Tuned LARGE LANGUAGE MODEL 3

Figure 2. Illustration demonstrates the use of one, or multiple, large language models to either: 1) fine-tune a large language model on proprietary domain-specific data, or 2) use a language model to further refine proprietary domain-specific data for use in downstream large language model fine-tuning. All steps can be securely and privately performed within a customer's infrastructure with Dell private cloud and/or on-premises platforms.

Dell Technologies PowerEdge XE9680 Server

The Dell PowerEdge XE9680 is a high-performance server designed and optimized to enable uncompromising performance for artificial intelligence, machine learning, and high-performance computing workloads. Dell PowerEdge is launching our innovative 8-way GPU platform with advanced features and capabilities.

- 8x NVIDIA® H100 80GB 700W SXM GPUs or 8x NVIDIA® A100 80GB 500W SXM GPUs
- 2x Fourth Generation Intel® Xeon® Scalable Processors
- 32x DDR5 DIMMs at 4800MT/s
- 10x PCIe Gen 5 x16 FH Slots
- 8x SAS/NVMe SSD Slots (U.2) and BOSS-N1 with NVMe RAID

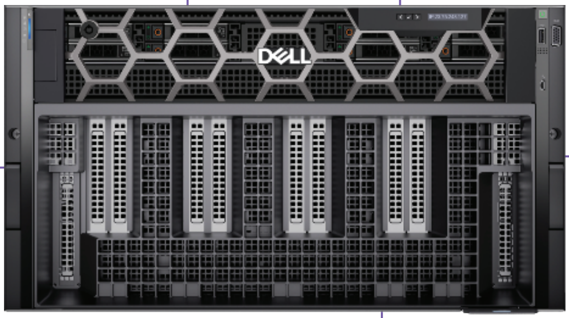
Dell PowerEdge XE9680

2 Socket Capable
- Up to two 4th Generation Intel® Xeon® Scalable processors with up to 56 cores per processor
- 6U air-cooled, up to 35C ambient
- 1200mm rack capable

Support for high-speed and memory capacity
- Up to 32 DDR5 DIMMs
- Up to 4800 MT/s (1DPC) or 4400 MT/s (2DPC)

AI

Large Model Training



I/O
- 10 x 16 PCIe Gen5 slots
- One OCP NIC 3.0
- 2 x 1GbE LOM

Support for up to 16 Drives
- Up to 8 x SAS/SATA/ NVMe Gen4 or 16x E3.S
- Rear Hot-Plug BOSS N -1 (2 x M.2 MVNe) for boot (optional)
- SW RAID/PERC12 support

GPU Optimized
- NVIDIA 8 x H100 SXM5 700W 80GB GPUs
  -or-
- NVIDIA 8 x A100 SXM4 500W 80GB GPUs
- Full NVLINK interconnectivity

	PowerEdge XE9680
CPU	2x Intel® Xeon® 8470 52-core Processor
GPU	8x NVIDIA® H100-SXM-80GB (700W)
System Memory	32x64GB – 2TB
Host NIC	NVIDIA® CX7

The PowerEdge XE9680 6U server is designed for AI, machine learning, and deep learning applications. It features the latest Intel® Xeon® processors with up to 56 cores, 8 NVIDIA® H100 or A100 GPUs, NVIDIA® NVLink™ technology for GPU-GPU communication, and supports up to 4 TB RDIMM of CPU RAM. The server supports virtualization options like NVIDIA® Multi-Instance GPU (MIG) capability, DDR5, NVLink™, PCIe Gen 5.0, and NVMe SSDs. It also supports NVIDIA® GDS (GPUDirect Storage), which provides a direct data path between GPU memory and storage, increasing system bandwidth and decreasing latency. The server is certified by NVIDIA® and has a secure, efficient, and comprehensive systems management solution with the OpenManage Enterprise console and iDRAC.

The results described in this article used a PowerEdge XE9680 server with 8 x 80 GB H100 NVIDIA® HGX GPU cards with NVLink™ technology, 2 TB of CPU RAM, and 2 x Intel® Xeon® processors on each server. The XE9680 server was configured with the Ubuntu v22.04 Linux operating system, Anaconda v23.1.0, CUDA v12.1, cuDNN v8.8.1, and the same python dependencies as noted by in the original report of the large language model fine-tuning benchmark.[17] A full list of the python dependencies installed on the XE9680 server can be found in the Appendix section of this article.

PowerEdge XE9680 Server Benchmarking Results

Dell Technologies has demonstrated large language models of comparable scale and performance to GPT-3 can be launched and queried on our PowerEdge XE9680 server platform.[19] Open-source language models include BLOOM-7B1,[13] GPT-J-6B,[15] and OPT-6.7B.[16] Large language models require sizable memory and GPU capabilities at inference and during model fine-tuning. In this instance, the open-source language models BLOOM-7B1, GPT-J-6B, and OPT-6.7B were launched, fine-tuned, and benchmarked on a PowerEdge XE9680 server.

Researchers at Stanford University recently demonstrated the benefits of using a large language model to further fine-tune and refine two existing large language models, Meta's LLaMA-7B[20] and OPT-6.7B[16] models, resulting in a new model known as ALPACA.[17] This process demonstrated a new paradigm in which the output of one large language model was used to fine-tune another large language model.

The Stanford researchers noted that their ALPACA model fine-tuning process required approximately 3 hours of continuous compute time on an undisclosed cloud computing system with a hardware configuration of 8 x 80 GB A100 NVIDIA® GPU cards, and an undisclosed amount of CPU RAM and I/O networking speeds.[21] Those same Stanford ALPACA benchmarks were run on a single Dell Technologies PowerEdge XE9680 server, using the same code and fine-tuning data set provided by the Stanford research team.[17] We were only able to compare our results against the OPT-6.7B language model, as the LLaMA language model was not open source. We did not have access to the LLaMA language model at the time this article was written, as access was restricted and regulated by Meta via an application process.[22]

Our PowerEdge XE9680 fine-tuning benchmarking results indicated a 10-to-12-fold reduction in fine-tuning time versus the Stanford cloud compute platform fine-tuning, with each process using eight GPUs (Figure 3). Our results are especially noteworthy as the PowerEdge XE9680 server significantly outperformed the cloud compute instance with the same number of GPUs.

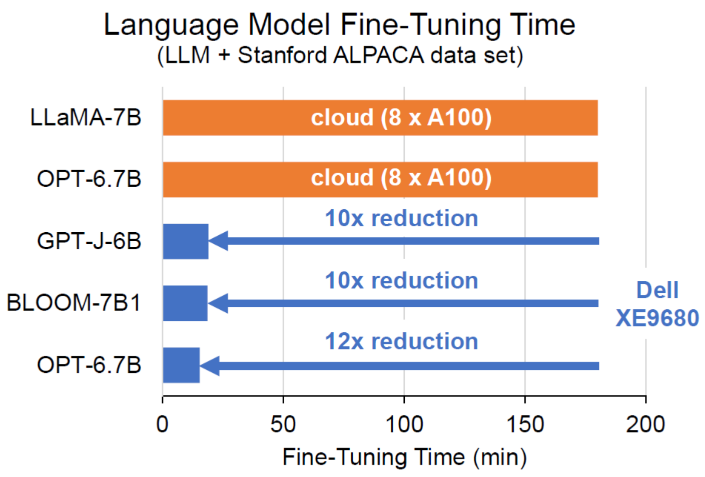


Figure 3. Language model fine-tuning time for various LLMs with the Stanford ALPACA data set, code, and process. Fine-tuning times for large language models (LLMs) on the Dell Technologies PowerEdge XE9680 server with 8 x NVIDIA® H100 80 GB GPUs (blue) vs. the unnamed cloud compute instance with 8 x NVIDIA® A100 80 GB GPUs (orange) using the same benchmarking data, process, and code. In this instance, the XE9680 demonstrated a greater than 10-fold reduction in OPT-6.7B language model fine-tuning time as compared with the OPT-6.7B fine-tuning on the cloud instance.

Impact of Data Set Size on Language Model Fine-Tuning

The impact of the data set size on the language model fine-tuning time was assessed using the OPT6.7B open-source language model.[16] The same benchmarking method as used above was applied in this survey,[17] and only the rows of fine-tuning data were varied. The data was the same set as utilized in the Stanford ALPACA study.[17] This data set contained 52,002 rows of data where each row contained a “question,” “input,” and “output” field. The 52,002 rows of data contained an average of 58.2 words per row of data, with a total of 3,025,587 words in the data set, and an average of 76.7 tokens per row of data, with 3,986,889 tokens in the data set (results based on OPT-6.7B tokenizer).[16] The smaller and larger data sets for this scaling study were assembled by sampling with replacement and a defined random seed for the discrete sample sizes show in Figure 4. A linear relationship was observed between the fine-tuning time of the language model and the rows of fine-tuning data used in the training process. The model fine-tuning process required approximately 3 minutes per 10,000 rows of fine-tuning data.

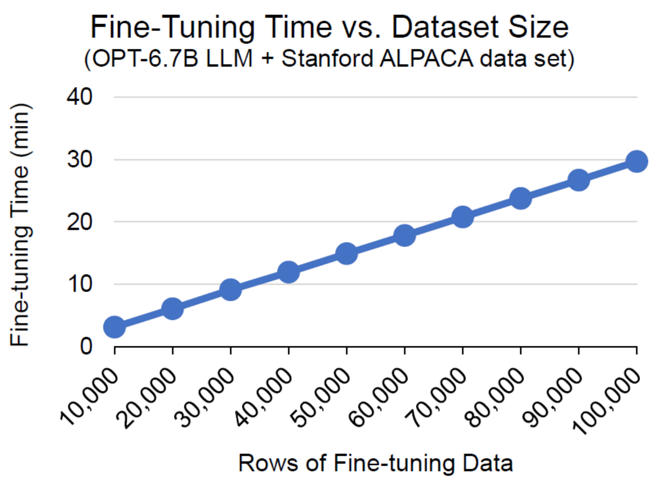


Figure 4. Language model fine-tuning time for various Stanford ALPACA fine-tuning data set sizes with the OPT-6.7B LLM and the Stanford ALPACA code and process on the Dell PowerEdge XE9680 server.

A larger-scale study of the data set size on the language model fine-tuning time was assessed using the OPT6.7B open-source language model.[16] The same benchmarking method as used above was applied in this survey,[17] and only the rows of fine-tuning data was varied over several orders of magnitude. The data sets for this larger scale study were assembled by sampling with replacement and a defined random seed for the discrete sample sizes show in Figure 5. For each entry in the resampled data set, a row corresponded to an average of 76.7 tokens, with the 10M row sample containing 766,798,807 tokens. Again, a linear relationship was observed between the fine-tuning time of the language model and the rows of fine-tuning data used in the training process, consistent with the results of the smaller-scale study in Figure 4.

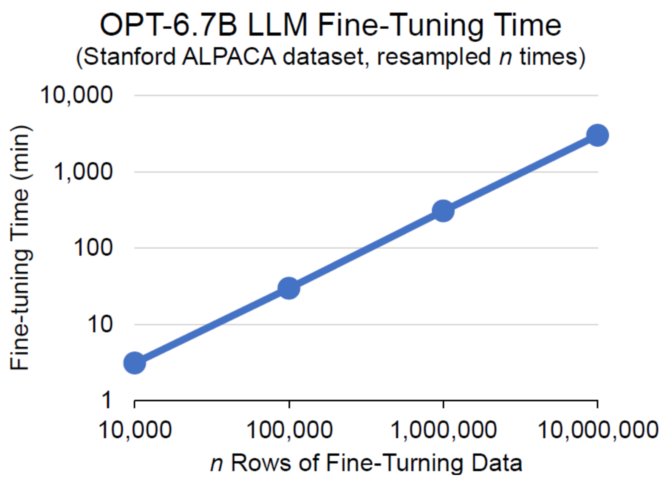


Figure 5. Large-scale study of language model fine-tuning time for various Stanford ALPACA fine-tuning data set sizes with the OPT-6.7B LLM and the Stanford ALPACA code and process on the Dell PowerEdge XE9680 server. The 10M row fine-tuning data set contained 767M tokens.

Impact of Language Model Size on Model Fine-Tuning

The impact of OPT language model size, by number of language model parameters, on the fine-tuning time was assessed with the Stanford ALPACA dataset, code, and process.[17] The OPT language model was available with 125 million, 350 million, 1.3 billion, 2.7 billion, 6.7 billion, 13 billion, and 30 billion model parameters.[16] OPT language models with 2.7 billion parameters, and fewer parameters, were fine-tuned on the Dell PowerEdge XE9680 server in less than 10 minutes each. The OPT-6.7B model required 15 minutes while the OPT-13B language model required 42 minutes to fine-tune. The larger OPT-30B language model resulted in out-of-memory (OOM) errors.

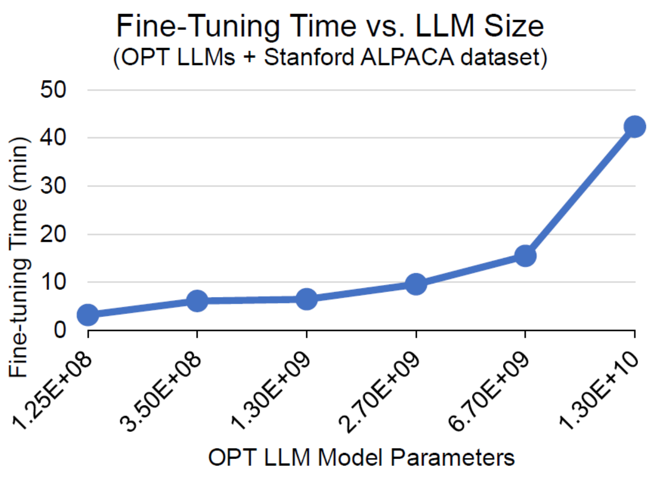


Figure 6. Fine-tuning times for OPT language models of varied sizes on the Stanford ALPACA fine-tuning data set, code, and process on the Dell Technologies PowerEdge XE9680 server.

Conclusion

In conclusion, large language models offer many advantages for general natural language processing tasks. However, practical use of these generalizations requires a fine-tuning process to create interfaces between the models and downstream applications to make them even more powerful. Various transfer learning approaches have been proposed to enable the fine-tuning of pretrained language models on specialized tasks and use cases. In-house machine learning expertise or trusted partners, high-performance computing hardware infrastructure, and structured datasets are needed to successfully implement this approach. Transfer learning is an effective strategy to customize large language models, but access to high performance computing resources and skilled talent is crucial, as this enables low latency responses to meet the performance requirements of an enterprise user base.

REFERENCES

1. Fauber, B. “Unleashing the power of large language models like ChatGPT for your business.” 2023, Dell Technologies white paper, https://www.delltechnologies.com/asset/en-us/solutions/infrastructure-solutions/industry-market/unleashing-the-power-of-large-language-models-fauber.pdf (accessed 11Apr2023).
2. Brown, et al. “Language models are few-shot learners.” Conference and Workshop on Neural Information Processing Systems Conference (NeurIPS) 2019.
3. Raina, R. et al. “Self-taught learning: transfer learning from unlabeled data.” International Conference on Machine Learning (ICML) 2007, 759–766.
4. “Catastrophic interference.” https://en.wikipedia.org/wiki/Catastrophic_interference (accessed 11Apr2023).
5. Chollet, F. “Deep learning with python.” 2017, Manning Publications, 384 pages.
6. Lee, J. et al. “BioBERT: A pre-trained biomedical language representation model for biomedical text mining.” Bioinformatics 2020, 36(4), 1234–1240.
7. Open AI Research. “ChatGPT.” 2022, https://chat.openai.com/
8. Ouyang, et al. “Training language models to follow instructions with human feedback.” 2022, arxiv.org/abs/2203.02155.
9. Houlsby, N. et al. “Parameter-efficient transfer learning for NLP.” International Conference on Machine Learning (ICML) 2019.
10. Hu, E. “LoRA: Low-rank adaptation of large language models.” International Conference on Learning Representations Conference (ICLR) 2022.
11. Gao, et al. “The Pile: An 800GB dataset of diverse text for language modeling.” 2020, arxiv.org/abs/2101.00027
12. Laurençon, et al. “The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset.” Conference and Workshop on Neural Information Processing Systems Conference (NeurIPS) 2022.
13. BigScience Workshop, et al. “BLOOM: A 176B-parameter open-access multilingual language model.” 2022, arxiv.org/abs/2211.05100
14. Muennighoff, N. et al. “Cross lingual generalization through multitask finetuning.” 2022, https://arxiv.org/abs/2211.01786
15. EleutherAI, et al. “GPT-J-6B.” 2021, https://huggingface.co/EleutherAI/gpt-j-6b (accessed 11Apr2023).
16. Facebook AI Research. “Democratizing access to large-scale language models with OPT-175B” 2022, https://ai.facebook.com/blog/democratizing-access-to-large-scale-language-models-with-opt-175b/
17. Taori, R. et al. “Stanford Alpaca: An instruction-following LLaMA model.” 2023, https://github.com/tatsu-lab/stanford_alpaca (accessed 11Apr2023).
18. Peng, et al. “Instruction tuning with GPT-4.” 2023, arxiv.org/abs/2304.03277.
19. Fauber and Patel. “Launching & running large language models on a single Dell server produces outstanding results.” 2023, Dell Technologies white paper, https://www.delltechnologies.com/asset/en-us/products/servers/industry-market/launching-llms-on-poweredge-xe9680.pdf (accessed 03May2023).
20. Meta AI Research. “Introducing LLaMA: A foundational, 65-billion-parameter large language model.” 24Feb2023, https://ai.facebook.com/blog/large-language-model-llama-meta-ai/
21. Taori, et al. “Alpaca: A strong, replicable instruction-following model.” https://crfm.stanford.edu/2023/03/13/alpaca.html (accessed 21Apr2023).
22. Meta AI Research. “LLaMA Request Form.” https://docs.google.com/forms/d/e/1FAIpQLSfqNECQnMkycAp2jP4Z9TFX0cGR4uf7b_fBxjY_OjhJILlKGA/viewform (accessed 21Apr2023)

APPENDIX

All results in this article were collected from a Dell Technologies PowerEdge XE9680 server. The server was configured with 8 x 80 GB H100 NVIDIA® HGX GPU cards with NVLink™ technology, 2 TB of CPU RAM, and 2 x Intel® Xeon® processors. The server was configured bare metal with the Ubuntu v22.04 Linux operating system, Anaconda v23.1.0, CUDA v12.1, cuDNN v8.8.1, and the same python dependencies as noted by in the original report of the large language model inference benchmark.[17] The python dependencies included torch v2.0.0+cu118, and transformers v4.26.1.

The Stanford ALPACA language model benchmarking code[17] was imported directly via git clone from https://github.com/tatsu-lab/stanford_alpaca. This data set contained 52,002 rows of data where each row contained a "question," "input," and "output" field. Each row contained an average of 58.2 words, with a total of 3,025,587 words in the data set, and an average of 76.7 tokens per row of data, with 3,986,889 tokens in the data set (results based on OPT-6.7B tokenizer).[16]

The language model benchmarking code was executed via the command line interface (CLI). The benchmarks were conducted in the same manner as described in reference 17. Benchmarks were conducted with all 8 GPUs available, and either the number of rows of fine-tuning data (10k, 20k, 30k, 40k, 50k, 60k, 70k, 80k, 90k, 100k, 1M, 10M) or the number of parameters in the OPT language model (125M, 350M, 1.3B, 2.7B, 6.7B, 13B, 30B) were altered. Only one variable was altered at a time. The language model fine-tuning time, as reported at the CLI after the process concluded, were reported as outputs.

As an example, the following CLI command was used for benchmarking the OPT-6.7B language model fine-tuning time on the ALPACA data set with 8 GPUs and the same default fine-tuning settings described by the Stanford ALPACA research team:[17]

```
torchrun --nproc_per_node=8 train.py \
   --model_name_or_path facebook/opt-6.7b \
   --data_path ./alpaca_data.json \
   --bf16 True \
   --output_dir=test_output \
   --num_train_epochs 3 \
   --per_device_train_batch_size 4 \
   --per_device_eval_batch_size 4 \
   --gradient_accumulation_steps 8 \
   --evaluation_strategy "no" \
   --save_strategy "steps" \
   --save_steps 2000 \
   --save_total_limit 1 \
   --learning_rate 2e-5 \
   --weight_decay 0. \
   --warmup_ratio 0.03 \
   --lr_scheduler_type "cosine" \
   --logging_steps 1 \
   --fsdp "full_shard auto_wrap" \
   --fsdp_transformer_layer_cls_to_wrap 'OPTDecoderLayer' \
   --tf32 True
```

Learn more about Dell solutions

Contact a Dell Technologies Expert

View more resources

Join the conversation with #PowerEdge #GenerativeAI