Balancing Size and Sustainability: The Role of Compression in Large Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

The rise of powerful Large Language Models like ChatGPT and GPT-4 has transformed AI across domains. However, their widespread use comes with significant environmental costs. To address this challenge, we propose a multi-pronged approach, including LLM compression, resource optimization, and active monitoring. This paper focuses on evaluating compression methods for sustainable LLM deployment in enterprise settings.

1 Introduction

The emergence of powerful Large Language Models (LLMs), like ChatGPT and GPT-4 [10, 11], along with accessible and cost-effective APIs, has accelerated the adoption of Generative AI across diverse domains. Companies like Hugging Face have further democratized AI by making LLMs accessible to novice programmers and software developers. Beyond the consumer society, startups such as Glean, Scale, Yurts, OpenAI, and Cohere are at the vanguard of a revolution, propelling generative AI into enterprises through innovative on-premises deployments and personalized large language models. These LLMs, rooted in the Transformer architecture [14], are meticulously designed for language comprehension and boast an extensive output vocabulary comprising English words. Their prowess stems from training on colossal text corpora, approaching a staggering 2 trillion words, endowing them with the remarkable ability to generate coherent and meaningful text. The moniker "Large Language Models" is not just a label; it aptly encapsulates the scale of these models, with more modest variants like BERT [3] and Flan [1] encompassing between 100 million and 500 million parameters, while behemoths like GPT-3.5 and GPT-4 push the boundaries with an astonishing 150 billion to 200 billion parameters.

While the widespread adoption of Large Language Models (LLMs) across domains is undeniably transformative, it brings to the forefront a pressing issue—the environmental cost of this technology's proliferation. Beyond the monetary expenses associated with deploying LLMs, there is a significant environmental burden to consider. Take, for example, the resources required to host a single model with 100B (billion) parameters (=400GB memory) for standard inference tasks. A 100B parameter model requires a minimum of 25 consumer GPUs, wherein each GPU has a capacity of 16GB. The number of models an organization needs to host depends on factors like the expected number of concurrent users and their tolerance for latency. For instance, accommodating up to 2 seconds of latency for 100 users would necessitate hosting one model per 100 users, translating to 10 instances of the same model for a generative AI experience serving 1000 concurrent users. With a single model requiring 25 GPUs, this equates to a staggering 250 GPUs running continuously. The environmental implications are substantial, as a single A4000 GPU running for a day emits approximately 1.24kg CO2-eq [6], equivalent to driving an internal combustion engine car for 5.1 km. Scaling this to the operation of 250 GPUs to serve 1000 users on the platform for a single day results in an expected emission of 3108kg CO2-eq! which is equivalent to a car journey spanning 12,500km, slightly

Submitted to Computational Sustainability Workshop at NeurIPS 2023. Do not distribute.

more than a round trip from California to New York. Balancing the benefits of LLM's with their environmental impact poses a critical challenge in the era of Generative-AI expansion.

In the pursuit of sustainable deployment of Large Language Models (LLMs) and generative AI applications, we advocate for a comprehensive, multi-pronged approach. Our strategy encompasses several key steps aimed at mitigating environmental impact while maintaining functional efficiency. First and foremost, we propose the development of mathematical methods to compress models to their sparser, low floating-point precision and lower parameter count counterparts of dense LLM's, while ensuring that their task performance remain uncompromised. Secondly, we tackle the scheduling challenge of efficiently allocating multiple models across minimal GPUs, introducing innovative sharding strategies to optimize resource utilization. Thirdly, we advocate a departure from a single, large foundational model to a mixture of smaller experts, significantly reducing GPU demands. Fourthly, we recommend the on-demand utilization of GPUs for hosting purposes and the implementation of predictive algorithms to estimate user traffic, minimizing the total GPU on-time. Lastly, we emphasize the importance of continuous monitoring, advocating for active tracking of GPU usage and its associated environmental costs within teams hosting LLMs and generative AI applications. This paper will focus on evaluating different compression methods for sustainable deployment of large language models, particularly when deployed at scale in enterprise settings.

2 Compression methods

Today, one of the most widely used commercial models is Meta's Llama-2-chat [13], introduced earlier this year. This model is pretrained on a massive dataset of 2 trillion tokens, roughly equivalent to about 500 billion words from the internet. Post pretraining, the model is further fine-tuned on multiple instruction datasets to enable it to provide precise answers to user queries (or user-instructions). The Llama-2 model comes in three different sizes: 7 billion parameters, 13 billion parameters, and 70 billion parameters, each demanding 28GB, 52GB, and 280GB of memory for their functioning. For hosting a single replica of the 7B, 13B, and 70B models, a minimum of 2, 4, and 18 consumer GPUs, (with each consumer GPU having upto 16GB memory) is required, respectively. In order to maintain low latency as the number of concurrent users increase, it would be essential to scale the number of model replicas. We find that the optimal scaling estimate would be to have a single replica of the model for every 100 concurrent users, ensuring that user latency remains minimal while minimizing the idle GPU time.

In this paper, we are specificially interested in assessing the role of model compression on the sustainable deployment of LLMs. We will focus on 3 aspects of compression.

- **Model quantization:** wherein the parameters of the model are stored in lower precision floating points (e.g. conversion from fp-32 to fp-16 or int-4).
- **Model sparsification:** wherein the non-zero parameters of the model are reduced, i.e. a large fraction of the models' parameters are set to zero.
- **Model distillation:** wherein a large model is used as a "teacher" to train a smaller "student" model for task-specific use-cases.

2.1 Model quantization

The standard deployment of any LLM is done in such a way that each parameter of the model is stored as a 32 bit floating point (fp-32), wherein the parameter value is stored using 8 exponent bits, 23 bits dedicated to the fraction and 1 bit for the sign (8+23+1=32 bits) [5], requiring 4 bytes per parameter. The estimates of model sizes provided in the earlier sections are arrived at by evaluating the product of 4 bytes per parameter with number of parameters (say, 7 billion) in the model (= 28GB). Quantization of models has been adopted as an effective method to quickly reduce the model size, across the ML industry. The common quantization levels used are: (i) 16 bits per parameter (fp-16), (ii) 19 bits per parameter (tfloat32), (iii) 16 per parameter (bfloat16), and (iii) 4 bits per parameter (int-4). That is, by quantizing the model to fp-16, or bfloat-16 the model size halves instantly, while quantizing to int-4 reduces the model size by 8 times! So, the Llama-2-7B model would require only 3.5GB of GPU memory while using an int-4 quantization and 14GB after fp-16 quantization, effectively reducing the number of GPUs required for hosting the model from 2GPU's to 1 GPU!

Also, depending on the hardware being used, the power consumption of fp16 and tfloat32 consume 5x lesser power than fp32, while bfloat-16 consumes 9x lesser power than fp-32!

Although quantization works well for some use-cases, in practice, it has been observed that the naive reduction of floating point precision for every parameter in the model (especially to int-4) may result in the steep decline of functional performance. New methods, like, Sparse Quantized Representations [2], Reparameterized Ternary network (RTN) [7], have been proposed to identify parameters in the model that are more robust to quantization, in order to not sacrifice the generative performance of the quantized LLM obtained.

2.2 Model sparsification

Model sparsification refers to reducing the number of non-zero parameters in the LLM. The rationale behind model sparsification is that the non-zero weights in the LLM require upto 4 bytes per parameter (fp-32), while the zeroed out weights can be expressed with a single bit of information. The ML community has developed a wide array of sparsification routines, for reducing the number of non-zero weights in a neural network (or LLM). Some of the well known methods are: (1) Lottery ticket hypothesis [4], (2) Iterative pruning and retraining of neural networks [9, 8], (3) traversing functionally invariant paths (FIPs) [12], to name a few. Each of these techniques discover sparser counterparts of dense networks while maintaining the network (or LLMs) functional performance.

We subjected BERT networks to traversal along FIPs [12] and discovered sparse BERTs that retain their high performance on general language understanding tasks [15] (e.g. QQP, QNLI), and also maintained their perplexity on the Wikipedia dataset, although being 50% sparser 1A, B!

We also calculated the memory footprint of a few popular open-source LLMs (like Llama-2, Falcon) post sparsification by converting the underlying regular matrix (with a large fraction of zeroed out parameters) that make up the LLM into a sparse architecture (into Pytorch COO or CSR formats). In 1C, we find that a single instance of dense Falcon-180B requires more than 40 consumer GPUs (16GB memory each), while its 60% sparser counterpart requires only half the number of GPUs ($\sim$20) for hosting a single instance. In Fig. 1D, we find that the CO2 equivalent for running a single instance of Llama-2-70B for a single day can be mitigated by sparsifying the model as well as by choosing to host these models in appropriate datacenters across the globe, as it was observed that hosting models on AWS datacenters in Asian countries produce a much larger CO2 equivalent when compared to the US west coast.

Although, sparse networks should require much lesser memory for their inference, we **do not** observe this in practice. We observe that sparse networks only have lesser storage memory requirements (while stored using sparse matrices on a hard-drive), but during inference, they tend to require a similar amount of online GPU memory while being deployed for inferences. At times, for matrices that aren't "sparse enough", the online GPU memory requirement of a sparse matrix is much larger than that of its denser counterpart.

To actually harness the reduced computational footprint afforded by a sparsified model, we would need to enhance our hardware and build out routines at the assembly level to enable computation with matrices of mixed precision types. That is, to enable a single matrix to have elements with different memory types (say, fp-16, int-4, binary). Without these updates, it would be difficult for us to take advantage of LLMs that have unstructured sparsity (i.e. the non-zero parameters aren't contiguous blocks within a matrix, or within the LLM). However, with our current hardware and assembly level implementations, LLMs with structured sparsity can still be taken advantage of for reducing the environmental burden of wide-scale deployment.

2.3 Model distillation

Compared to the above mentioned methods of compressing models, the method of model distillation is significantly different, as it uses the original larger language model as the "teacher" model for training a much smaller "student" LLM (with any network architecture), on a task-specific use case. In this scenario, a synthetic dataset of inputs and target outputs are generated from the "teacher" LLM, which is subsequently used for training the student LLM via the supervised learning paradigm.

Distillation is extremely effective for building task-specific (smaller) models (or specialized experts). In many enterprise deployment settings, we find that the natural language tasks are very pointed,

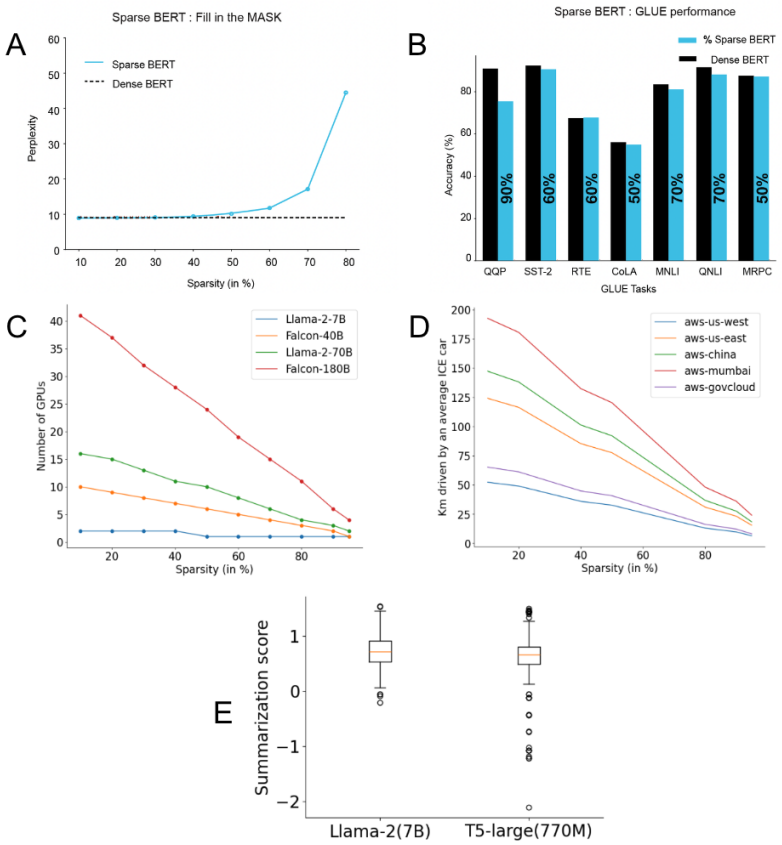

Figure 1: **Role of compression in sustainable deployment** (A) Perplexity score for sparse BERT (solid blue) and dense BERT (dashed blue). (B) GLUE performance of dense BERT (black) and p% sparse BERT (blue). (C) Quantifying memory requirement (in terms of GPU count) for a single replica of well known open-source models. (D) Quantifying CO2 emissions in terms of Km driven by an average ICE car [6] for hosting a single Llama-2-70B model for 24 hours. (E) Summarization score of the "teacher" Llama-2 (7B) and "student" T5-large (770M), post model distillation, on an evaluation dataset.

(for instance, summarization of an article), which makes distillation very effective. In Fig. 1E, we observe that, distillation can be effectively used to shrink a Llama-2-7B (with 7B parameters) to a T5-large (770M parameters) for a specific task like summarization without incurring a massive loss in functional performance.

3 Discussion

In this paper, we've explored vital avenues for compressing large language models (LLMs) to enable their sustainable deployment. Model quantization, sparsification, and distillation offer promising ways to reduce memory and computational demands. Model quantization, achieved through lower-precision formats, can significantly cut down model size and power consumption. However, a balance must be struck to avoid sacrificing performance, necessitating advanced techniques. Sparsification of LLMs can, theoretically speaking, lower GPU requirements and decrease CO2 emissions. However, it currently doesn't reduce GPU memory usage during inference, highlighting the need for more development on the hardware and assembly level routines. Model distillation, where smaller models learn from larger ones, proves effective for task-specific applications, but aren't the best choice for preserving the ability to perform a wide array of tasks. Choosing the right compression technique depends on specific use cases and resource constraints. Future research and hardware improvements will further enhance LLM deployment efficiency, ensuring their power is harnessed responsibly.

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[2] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[5] William Kahan. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996.

[6] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[7] Yuhang Li, Xin Dong, Sai Qian Zhang, Haoli Bai, Yuanpeng Chen, and Wei Wang. Rtn: Reparameterized ternary network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4780–4787, 2020.

[8] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 33:2589–2604, 2021.

[9] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.

[10] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[12] Guruprasad Raghavan and Matt Thomson. Engineering flexible machine learning systems by traversing functionally invariant paths in weight space. *arXiv preprint arXiv:2205.00334*, 2022.

[13] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.