Towards Green Small Language Models

Anonymous Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymouser Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymousest Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

Rapid advances in natural language processing have led to the development of large language models (LLMs), which are based on the Transformer architecture. These models have grown significantly in size, and their increasing demand for computational resources has raised concerns about their environmental impact. The substantial carbon footprint and the water footprint during training and inference contribute to global warming, highlighting the need for more energy-efficient approaches. Our study attempts to address these challenges by compressing the size of LLMs. Our approach lies in reducing the dimensionality of the model parameter space using a well-known dimensionality reduction technique such as Singular Value Decomposition and Autoencoders. Specifically, we apply these dimension reduction techniques to the large weight matrices of LLMs. We target these matrices as they contain parameters, encoding knowledge learnt by the LLMs. Thus, we reduce the LLMs' size, which should positively impact its size, speed, resource-consumption, and overall, its carbon footprint. In this paper, we present our initial results on Llama-2 with 7 billion parameters.

1 Introduction

Large language models (LLMs) have greatly improved the field of Natural Language Processing (NLP), showing strong abilities in tasks like text generation, language understanding, and translation. These models are trained (Hoffmann et al., 2022) on vast datasets from various sources, allowing them to learn complex language patterns. However, training LLMs requires vast computational resources, often involving the use of specialized hardware over extended periods, resulting in considerable power consumption. This substantial energy consumption raises critical concerns about environmental impact of LLM (Li et al., 2023) (Li et al., 2024) (Crawford, 2024). The carbon footprint (Luccioni et al., 2023) associated with training and inference processes has become a key issue, especially as the size and complexity of models continue to increase. For instance, the pretraining of the Llama 3 model (Dubey et al., 2024) utilized 7.7 million GPU hours on H100-80GB hardware (700W), generating an estimated carbon emission of 2290 tCO2eq. In response, researchers are focusing on developing smaller models (Allal et al., 2024) (e.g., smolLLM, Llama3.2-1B, Qwen2.5-1B) or compressing large models (e.g., GPT-4-1.8T, Llama3.1-8B/70B/405B, Mistral-8B/7x7B/8x22B) into smaller ones (Gu et al., 2024) (Cheng et al., 2024) for deployment in resource-constrained environments. Training large language models is energy-intensive, with Hugging Face reporting (de Vries, 2023) that its BigScience Large Open-Science Open-Access Multilingual (BLOOM) model consumed 433 MWh and Llama-3 5,390 MWh, while other models like GPT-3, Gopher, and Open Pre-trained Transformer (OPT) (Zhang et al., 2022) required 1,287 MWh, 1,066 MWh, and 324 MWh, respectively. These figures highlight the urgent need for research to reduce LLMs environmental impact while preserving performance. Our study addresses this challenge by investigating the application of classical dimensionality reduction techniques, such as Singular Value Decomposition (SVD) (Wang and Zhu, 2017) and autoencoders (Chen and Guo, 2023), to reduce model parameter dimensions and LLM size. We describe our motivation for adopting such a strategy in the next section.

2 Background & Motivation

Large language model parameters (Roberts et al., 2020) are numerical values (e.g., weights) that impact the model training and inference. These parameters, which range from millions to trillions in number, are strategically distributed across the LLM architecture (Wang et al., 2022) to capture complex patterns, relationships, and structures in the data. They are typically organized into distinct components of the architecture. In the input layer, parameters initialize embeddings that convert raw input (e.g. tokens) into numerical vectors, capturing semantic and positional information. Within the hidden layers, parameters in Transformer blocks enable attention mechanisms and feed-forward networks to process and refine the input, identifying meaningful relationships and hierarchical structures. Finally, in the output layer, parameters map these processed representations back into the target space, such as generating probabilities for token prediction. The scale and distribution of these parameters directly impact the model ability to generalize and adapt to a wide range of tasks. However, more parameters increases computational requirements and energy consumption.

3 Methodology

Some parameters that capture noisy or irrelevant data can lead to undesirable phenomena, such as hallucination (Ji et al., 2023) or result in minimal contributions to the model overall performance. Empirical studies on pruning (Cheng et al., 2024), (Ma et al., 2023) have shown that parameters with low importance scores, below a predefined threshold, can be removed with negligible impact on model performance. Our study focuses on the knowledge contained within the LLM parameters. Our premise is that not all patterns captured by the parameters are important for a specific task. Our approach therefore involves reducing the parameter space, preserving only the most relevant patterns. We seek to reduce the number of parameters and ultimately to lower the carbon footprint of LLM. We achieve this goal using Singular Value Decomposition and autoencoders, applied to the weight matrices. Our approach offers a trade-off between linear and non-linear compression strategies, allowing for efficient compression without significant loss in model performance.

3.1 Framework

Given a pre-trained LLM, we fine-tune (Anisuzzaman et al., 2024) it for a specific task and then apply the proposed dimensionality reduction method to obtain a reduced version of the model, as illustrated in Figure 1. In the decoder-

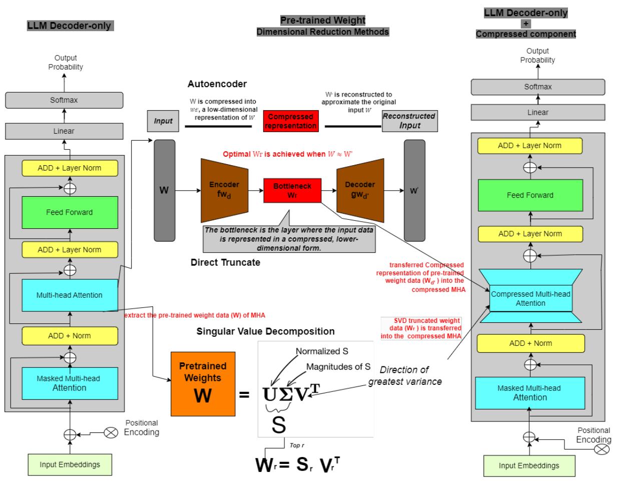
LLM Decoder-only
Pre-trained Weight
Dimensional Reduction Methods
LLM Decoder-only
+
Compressed component
Output Probability
Softmax
Linear
ADD + Layer Norm
Feed Forward
ADD + Layer Norm
Multi-head Attention
ADD + Norm
Masked Multi-head Attention
Positional Encoding
Input Embeddings
Autoencoder
Input
W is compressed into Wd', a low-dimensional representation of W
Compressed representation
W' is reconstructed to approximate the original input W
Reconstructed Input
Optimal Wr is achieved when W ≈ W'
W
Encoder fwd
Bottleneck Wr
Decoder gwd'
W'
The bottleneck is the layer where the input data is represented in a compressed, lower-dimensional form.
Direct Truncate
transferred Compressed representation of pre-trained weight data (Wd') into the compressed MHA
extract the pre-trained weight data (W) of MHA
Singular Value Decomposition
SVD truncated weight data (Wr) is transferred into the compressed MHA
Pretrained Weights W
=
Normalized S
Magnitudes of S
UΣV^T
Direction of greatest variance
S
Top r
Wr = Sr Vr^T
Compressed Multi-head Attention

Figure 1: Integration of autoencoders and SVD into a LLM based decode-only .

only architecture (Wu et al., 2023), we focus on the Multi-Head Self-Attention (MHA) layer (Vaswani, 2017), which involves three linear transformations that project the input embeddings into query, key, and value spaces. The attention output is obtained through a fourth transformation applied to the weighted sum of the values. Our

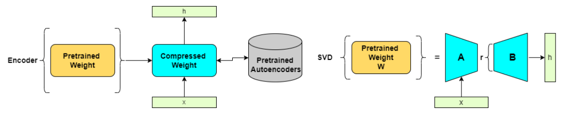

Figure 2: Our reparametrization with Autoencoders (left) and SVD (right).

reparametrization is illustrated in Figure 2. The pre-trained weights of the LLM are either replaced with their compressed representations using autoencoders or decomposed into two matrices based on singular value decomposition (SVD) and rank. Our proposed approaches using SVD and autoencoders are presented in Algorithm 1 and Algorithm 2, respectively.

4 Experimental Result

In this section, we present the initial results of our ongoing experiments.

Algorithm 1 Extraction of MHA Weights and SVD Computation

```
1: Input: Weight matrices $W_q, W_k, W_v, W_o$ from MHA layer
2: Parameters: Magnitude Threshold $T_{\text{min}}$, rank $r$
3: Step 1: Compute the Singular Value Decomposition (SVD) for each weight matrix
       $$W = U\Sigma V^T$$
4: Step 2: Truncate $\Sigma$ based on singular value magnitude: retain $r$ singular values where $\sigma_i \geq T_{\text{min}}$
5: Step 3: Construct matrices $A$ and $B$:
       $$A = U_r\Sigma_r \text{ and } B = V_r^T$$
6: Step 4: Reconstruct the low-rank approximation:
       $$W_r = AB$$
7: Output: Low-rank approximations of the weight matrices $W_q, W_k, W_v, W_o$
```

Algorithm 2 Dimensionality Reduction of MHA Weights using Autoencoder

```
1: Input: Weight matrices $W_q, W_k, W_v, W_o$ from the MHA layer, pre-trained autoencoder for weight matrix reconstruction with latent space dimension $r$.
2: Step 1: Encode each weight matrix into the latent space and reconstruct it from the latent representation:
       $$W_r = \text{Encoder}(W), \quad W' = \text{Decoder}(W_r)$$
3: Step 2: Replace the original pre-trained weight matrices with their latent representations in memory. Store the autoencoder model for future use during inference.
4: Inference: During inference or fine-tuning, the latent representations are reconstructed using the decoder to approximate the original weights, then re-compressed and stored in memory for efficient use.
```

4.1 Experimental Settings

Our proposed method is applied to the multi-head attention component of LLaMA-2 with 7 billion parameters (Touvron et al., 2023). The autoencoder architecture used to compress the weight parameters is a two hidden layer convolutional neural network with a total of 2434 parameters. For both autoencoders and SVD, the rank is set to one-quarter of the embedding size. The multi-head attention (MHA) layer of LLaMA-2 7B has 4 weight matrices and 32 attention heads, resulting in a total of 128 parameters. Our autoencoder training configuration includes a batch size of 128, a learning rate of 1e-4, and a total of 50 epochs. We used the parameters of the first 8 heads (32 parameters) for testing and the last 24 heads (96 parameters) for training, based on the assumption that the later heads capture more patterns during the LLM training phase. We evaluated the autoencoder using the Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) (Bakurov et al., 2022) metrics. All experiments are conducted using an NVIDIA A100 80GB GPU provided by the Consortium des Équipements de Calcul Intensif (CÉCI) [1].

5 Results

Autoencoder: The experimental results in Figure 3 show that the autoencoder achieves low reconstruction error, as indicated by the MSE on the order of 10^{-3}, and a steady improvement in SSIM scores with increasing epochs. This demonstrates

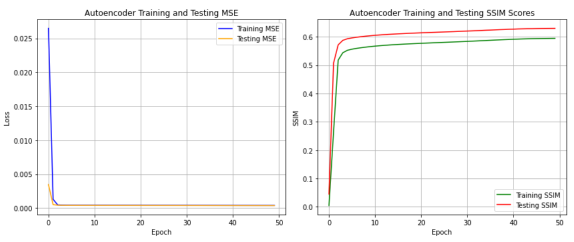

Figure 3: **Left**: Mean Square Error (MSE) of the pre-trained weight and its reconstruction. **Right**: Structural Similarity Index Measure (SSIM) for Reconstructed weight Quality Assessment.

that the reconstructed weights closely resemble the original weights, reflecting the autoencoders effectiveness. Table 1 further highlights a significant reduction in the total parameter count from

[1]https://www.ceci-hpc.be/

7B to approximately 5B after compression. The most notable reduction occurs in the Multi-head Attention component, where parameters decrease

Component	LLaMA-2-7B Parameters	Compressed Parameters
Multi-head Attention	2,147M	**134M**
Feed-Forward Neural Network	4,329M	4,329M
Others (embedding layer, etc.)	262M	262M
Autoencoders	-	2434
Total	$6,738M \approx 7B$	$4,725M \approx 5B$

Table 1: Comparison of LLaMA-2-7B parameter sizes before and after compression using autoencoders.

from 2,147M to 134M, demonstrating the efficiency of the autoencoder. This reduction showcases the ability of the proposed method to effectively compress the LLaMA-2-7B model while preserving its structural integrity and reconstruction quality.

Singular Value Decomposition: The results presented in Table 2 demonstrate the effectiveness of Singular Value Decomposition (SVD) in compressing the LLaMA-2-7B model. By applying

Component	LLaMA-2-7B Parameters	Compressed Parameters
Multi-head Attention	2,147M	**1,073M**
Feed-Forward Neural Network	4,329M	4,329M
Others (embedding layer, etc.)	262M	262M
Total	$6,738M \approx 7B$	$5,664M \approx 6B$

Table 2: Comparison of LLaMA-2-7B parameter sizes before and after compression using SVD with rank = (embedding size/4).

SVD with a rank of embedding size/4, the total parameter count is reduced from 7B to approximately 6B. The most significant reduction occurs in the multi-head attention component, where the parameters decrease from 2,147M to 1,073M, reflecting the efficiency of SVD in reducing dimensionality. This result highlights the potential of SVD as a compression method to achieve parameter reduction while maintaining key model components.

6 Discussion

The experimental results reveal the potential of Autoencoders and Singular Value Decomposition (SVD) as effective techniques for compressing large-scale language models like LLaMA-2-7B. Autoencoders achieve significant parameter reduction by leveraging non-linear transformations, making them suitable for aggressive compression. In contrast, SVD offers a simpler, linear method with moderate compression, maintaining a balance between efficiency and interpretability. These results are preliminary and represent the first steps in an ongoing study. Further experiments are necessary to evaluate the trade-offs between memory reduction, carbon footprint, and the impact of information loss on model performance. Additionally, the computational cost of the compression process, particularly with Autoencoders, must be analyzed to ensure the overall energy efficiency of the approach. Future work will focus on optimizing these methods to maintain the quality of downstream tasks while achieving meaningful reductions in model size and resource consumption.

7 Conclusion

This work explores the integration of Autoencoders and Singular Value Decomposition (SVD) as techniques for compressing large-scale language models, such as LLaMA-2-7B. Preliminary findings highlight the effectiveness of both methods in reducing model size, with Autoencoders achieving higher compression ratios and SVD providing a simpler yet effective alternative. While the results demonstrate the potential for memory and parameter reduction, further evaluations are required to assess their impact on downstream tasks, performance trade-offs, and energy efficiency. This ongoing research aims to optimize these techniques to create more sustainable and resource-efficient AI models without compromising functionality or accuracy.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. 2024. Smollm-blazingly fast and remarkably powerful. *Huggingface Blog*.

DM Anisuzzaman, Jeffrey G Malins, Paul A Friedman, and Zachi I Attia. 2024. Fine-tuning llms for specialized use cases. *Mayo Clinic Proceedings: Digital Health*.

Illya Bakurov, Marco Buzzelli, Raimondo Schettini, Mauro Castelli, and Leonardo Vanneschi. 2022. Structural similarity index (ssim) revisited: A data-driven approach. *Expert Systems with Applications*, 189:116087.

Shuangshuang Chen and Wei Guo. 2023. Auto-encoders in deep learning—a review with new perspectives. *Mathematics*, 11(8):1777.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. Mini-llm: Memory-efficient structured pruning for large language models. *arXiv preprint arXiv:2407.11681*.

Kate Crawford. 2024. World view: Generative ai's environmental costs are soaring and mostly secret. *Nature*, 626:693.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Sprout: Green generative ai with carbon-efficient llm inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21799–21813.

Pengfei Li, Jianyi Yang, Mohammad A Islam, and Shaolei Ren. 2023. Making ai less" thirsty": Uncovering and addressing the secret water footprint of ai models. *arXiv preprint arXiv:2304.03271*.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2023. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Alex de Vries. 2023. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization? In *International Conference on Machine Learning*, pages 22964–22984. PMLR.

Yongchang Wang and Ligu Zhu. 2017. Research and implementation of svd in machine learning. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 471–475. IEEE.

Jian Wu, Yashesh Gaur, Zhuo Chen, Long Zhou, Yimeng Zhu, Tianrui Wang, Jinyu Li, Shujie Liu, Bo Ren, Linquan Liu, et al. 2023. On decoder-only architecture for speech-to-text and large language model integration. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.