ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ARTIFICIAL INTELLIGENCE

Optimizing Small Language Models: An Experimental Investigation For Compressing Distilled LLaMa-based Models

Supervisor:
Prof. Francesco Conti

Presented by:
Angelo Galavotti

Co-supervisor:
Luca Bompani

Session I
Academic Year 2024/2025

Alla migliore madre del mondo,
al miglior padre del mondo.

Abstract

Pending

Contents

List of Figures

Chapter 1

Introduction

It is no secret that, in the last three years, *Large Language Models* (LLMs) have fundamentally transformed our relationship with technology. Their impact rivals the most significant innovations of the past century, such as the internet and the smartphone. When people contemplate *Artificial Intelligence* (AI) today, they immediately think of ChatGPT or Claude, which have seamlessly integrated into our daily routines. Yet these powerful tools come with significant environmental concerns. Their development and operation consume vast amounts of energy and water resource: modern data centers supporting these models require extensive cooling systems and electricity consumption that can rival small cities.

The computational complexity of these systems necessitates cloud-based deployment, which not only amplifies their environmental footprint but also fundamentally restricts user autonomy. This cloud dependency creates a concerning power dynamic where users have limited control over their tools, while simultaneously enabling extensive data collection practices and potential surveillance mechanisms that would be impossible with local, user-controlled alternatives.

In this more conversational opening chapter we will briefly examine the environmental and social impact of LLMs while highlighting the growing imperative for efficient, locally-deployable models that democratize access without depleting our planet's resources. Afterwards, we will outline the scope of this project, which aims to explore the potential of compression techniques to make LLMs more efficient.

The future of AI depends not just on what these models can do, but how sustainably they can do it.

1.1 The social impact of LLMs

The widespread adoption of LLMs has created ripple effects across virtually every sector of society, fundamentally altering how we work, learn, and create. In education, these tools have sparked heated debates about academic integrity while simultaneously offering new possibilities for personalized learning and accessibility for students with disabilities [1]. The workplace has experienced perhaps the most dramatic shifts, with entire professions grappling with automation anxiety while others discover unprecedented productivity gains. Creative industries find themselves in particularly complex territory: writers, artists, and content creators must navigate between leveraging AI as a collaborative tool and protecting their intellectual property from being absorbed into training datasets without consent or compensation [2].

Perhaps what is most striking is how these models have democratized access to sophisticated capabilities that were once the exclusive domain of experts. A small business owner can now generate marketing copy that rivals professional agencies, students can receive tutoring in subjects where human expertise might be scarce, and non-programmers can write functional code with natural language instructions. Yet this democratization comes with a troubling caveat: it's entirely dependent on maintaining access to centralized, corporate-controlled systems. When OpenAI experiences an outage, millions of users worldwide suddenly lose access to tools they've integrated into their daily workflows. When pricing models change, entire business models built around AI assistance can become unsustainable overnight.

This dependency becomes even more concerning when we consider the data these systems collect. Every interaction, every query, every creative prompt may potentially become part of these companies' datasets, raising questions about privacy and intellectual property. As such, the need for transparency and user control over these systems has never been more urgent, and many believe that the future of AI must prioritize local, user-deployable models that empower individuals rather than centralizing power in the hands of a few corporations.

1.2 The environmental impact of LLMs

The computational demands of modern LLMs create an environmental footprint that grows exponentially with model size and usage. Training GPT-3, for instance, consumed an estimated 1,287 MWh of electricity, which is enough to power an average American home for over a century [3]. However, training represents only the tip of the iceberg; the real environmental cost lies in inference, where billions of daily queries across millions of users create a continuous drain on global energy resources. A recent study by Jegham et al. [5] estimates that OpenAI's GPT-4.5 requires 6.7 Wh of energy for a medium sized query (i.e. 100 input tokens, and outputting 300 tokens) to be processed. This figure grows to 20.5 Wh for a larger query (i.e. 1000 input tokens, and outputting 1000 tokens). To put this into perspective, this is equal to charging an average 40 Wh laptop battery to 50%. A graph comparing the energy consumption of different LLMs with different prompt sizes is shown in Figure 1.1.

Data centers housing these models consume approximately 1-2% of global electricity, a figure that's projected to reach 8% by 2030 if current trends continue [4]. The infrastructure supporting a single large-scale LLM requires thousands of high-performance GPUs running 24/7, each consuming as much power as several households.

Water consumption presents an equally pressing concern that receives far less attention. Modern data centers require extensive cooling systems, with some facilities consuming millions of gallons daily. According to their sustainability report [6], Google's water usage increased by 20% between 2021 and 2022, then by 17% from 2022 to 2023, and is largely attributed to AI inference operations [7]. In regions already facing water scarcity, this additional demand creates direct competition with human needs and agricultural requirements.

On the other hand, one has to keep in mind the embedded carbon cost of the hardware itself. Each GPU cluster supporting LLM operations represents significant emissions from manufacturing, shipping, and eventual disposal. The rapid pace of AI advancement drives frequent hardware upgrades, creating electronic waste streams that the recycling industry struggles to process effectively.

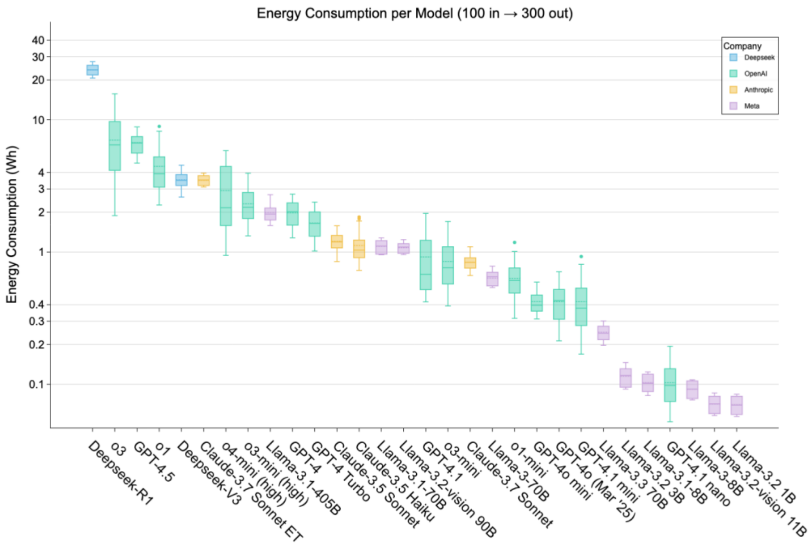
These environmental costs scale directly with model size and usage frequency, creating a fundamental tension between AI capabilities and sustainability.

1.3 Scope

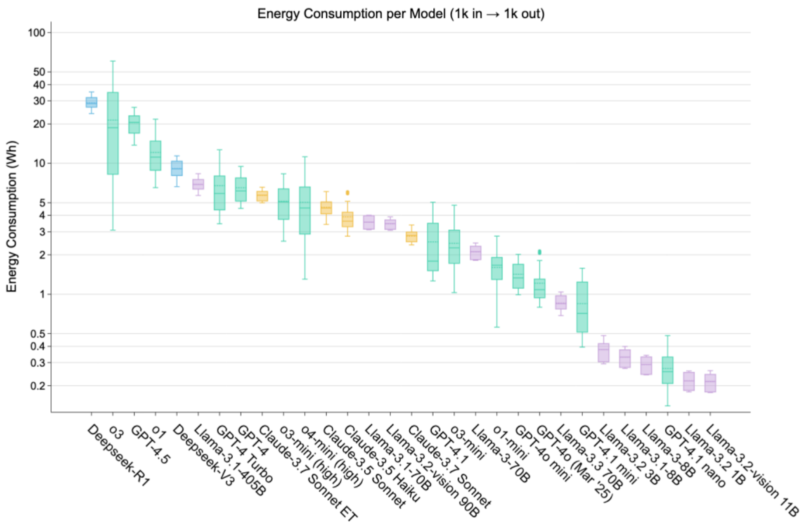
While we've examined several critical challenges facing LLMs, it's worth emphasizing that these models hold significant potential for positive impact. As such, what was outlined in the previous sections points instead toward an urgent need for alternatives to the current paradigm of massive, centralized LLMs. A promising solution lies in compression techniques that can dramatically reduce model size while preserving core functionality. Through compression, billion-parameter server-sized models can be scaled down to more manageable ones that run much more efficiently.

1.3.1 The main objective

Given the context, the objective of this project is to push the boundaries of memory efficiency for small-scale LLMs by applying a targeted suite



(a) Energy consumption for a small query (100 input tokens, 300 output tokens).



(b) Energy consumption for a large query (1000 input tokens, 1000 output tokens).

Figure 1.1: Comparison of the energy consumption of different LLMs for small and large queries. The data highlights the significant increase in energy requirements as query size grows. These graphs have been sourced from [5].

of compression techniques. We begin with a distilled model, a 1B parameter variant of LLaMA 3 [9], which already represents a significantly reduced footprint compared to full-scale LLMs. From there, we explore and implement further optimizations, including depth and width pruning, *Low-Rank Adaptation* (LoRA), and quantization.

Pruning allows us to remove redundant layers or neurons from the network architecture, trimming excess capacity without substantial loss of capability. Quantization reduces the bit-width of model weights and activations, decreasing both memory usage and compute requirements. LoRA, meanwhile, introduces a lightweight, parameter-efficient training mechanism that reduces the cost of fine-tuning and adaptation without (necessarily) modifying the base model weights. In this way, the model can be adapted to new tasks or domains with minimal overhead.

The reason behind the focus on maximizing memory efficiency is related to the fact that memory represents the most expensive constraint in our target deployment scenario, where the intended hardware platform consists of a low-power RISC-V SoC with severely constrained memory resources (further details on the target hardware can be found in Section 2.4). Crucially, optimizations that reduce memory footprint also translate directly into computational complexity improvements, creating a dual benefit for resource-constrained environments.

1.3.2 How can optimization techniques help?

Optimization techniques tackle the environmental, social, and infrastructure problems that come with large-scale LLMs. When models run more efficiently, they need far less power for each query. Smaller models can actually run on edge devices and other energy-efficient hardware, cutting down on electricity usage substantially. This efficiency boost also extends the useful life of older devices: hardware that might otherwise be considered obsolete can suddenly run modern LLMs, which helps reduce electronic waste and makes better use of existing resources.

There's also the autonomy angle. Compact models that run locally allows users to be independent from centralized servers when using AI. People can run LLMs right on their own devices without any internet connection, which means no data gets sent to third parties and there's no risk of surveillance. This approach supports decentralization and opens up AI access to areas with poor connectivity or limited economic resources.

In other words, optimization is a pathway toward environmentally sustainable, privacy-respecting, and widely accessible AI.

1.4 Document structure

Will write this when the document is finished.

Chapter 2

Background and Related Work

Before explaining the details and implementation of the methodology used in the context of this thesis, it is essential to provide an overview of the evolution of the inner workings of the Transformer architecture as well as Language Models in general. In addition, in this chapter covers relevant optimization techniques and how they influenced the final result. Finally, we will also shed some light on the target hardware, whose limitations have been a driving force behind the design choices made in this project.

2.1 Early Language Models

Before the emergence of the Transformer architecture, language models predominantly relied on *Recurrent Neural Networks* (RNNs) . These networks represent an evolution of the *Multi-Layer Perceptron* (MLP), incorporating cyclic connections within their architecture to create recurrent circuits. This design enables the network to maintain a form of memory, allowing it to consider previous inputs when generating predictions and thereby capturing long-range dependencies inherent in sequential data such as text.

Despite their theoretical advantages, RNNs suffer from a critical limitation known as the vanishing gradient problem [12]. During backpropagation, throughout time, gradients from earlier steps diminish exponentially as they propagate backward through the network layers. This degradation severely hampers the network's ability to be trained effectively, as the influence of distant past information becomes negligible in the parameter updates.

To address these limitations, Hochreiter and Schmidhuber introduced *Long Short-Term Memory* (LSTM) networks [13], which revolutionized sequence modeling through their sophisticated gating mechanisms. LSTMs employ specialized memory cells designed to selectively retain, update, and output information across extended sequences. The architecture incorporates three fundamental gate types that regulate information flow:

- The *input gate* determines the extent to which new information from the current input should be incorporated into the cell state. It evaluates the relevance of incoming data and controls its integration with existing memory.
- The *forget gate* governs the retention of information from previous time steps, deciding which aspects of the historical cell state remain relevant and should be preserved.
- Finally, the *output gate* regulates how much of the current cell state should be exposed to subsequent layers, effectively controlling what information propagates forward in the network.

This gating mechanism allows LSTMs to maintain gradient flow over much longer sequences compared to vanilla RNNs. Thus, they can capture dependencies spanning hundreds of time steps, making them particularly effective for tasks requiring long-term context understanding such as language modeling, machine translation, and text summarization. A visual representation of an LSTM cell is shown in Figure 2.1.

While LSTMs significantly improved the ability to model sequential data, they still faced challenges in terms of parallelization and context understanding, especially when compared to Transformers. Nevertheless, LSTMs are still widely used for various *natural language processing* (NLP) tasks, including text generation [14].

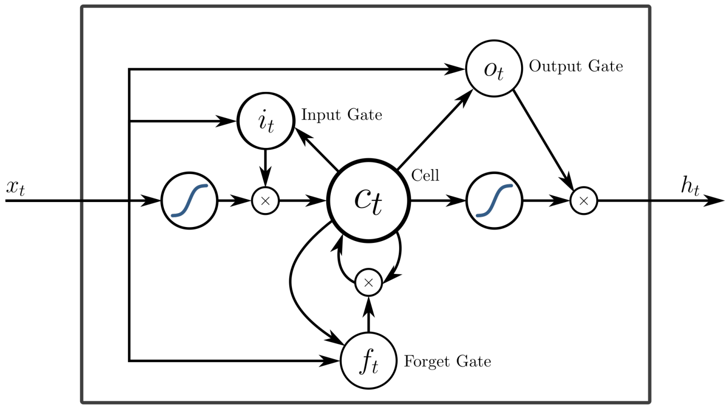


Figure 2.1: Architecture of an LSTM cell showing the three gating mechanisms. The input gate (left) controls information incorporation, the forget gate (center) manages memory retention, and the output gate (right) regulates information flow to subsequent layers. The cell state (horizontal line) maintains long-term memory throughout the sequence.

2.2 The Transformer Architecture

The Transformer architecture, introduced by Vaswani et al. [15], fundamentally changed how we approach sequence modeling by abandoning recurrent connections entirely in favor of attention mechanisms. Unlike LSTMs that process sequences step-by-step, the Transformer can examine all positions simultaneously, allowing for efficient parallelization during training. Moreover, it can capture long-range dependencies thanks to the attention mechanism.

2.2.1 The Attention Mechanisms

At the core of the Transformer lies the attention mechanism, which fundamentally changed how neural networks process sequential information. The attention mechanism addresses the information bottleneck created when sequence models compress an entire input sequence into a single fixed-size vector. This bottleneck becomes particularly problematic for long sequences, where important information can be lost or diluted.

Attention allows a model to dynamically focus on different parts of the input sequence when generating each output token, similar to how humans selectively concentrate on relevant information when reading or translating text. This is done, essentially, by directly comparing and relating any two positions in a sequence, regardless of their distance (i.e. self-attention). This selective focus mechanism enables the model to capture long-range dependencies more effectively and provides interpretable insights into which input elements influenced specific predictions.

2.2.2 Scaled Dot-Product Attention Mechanics

The Transformer's scaled dot-product attention operates on three matrices derived from the input:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Each component serves a specific purpose:

- **Queries (Q)**: Generated by $Q = XW_Q$ where X is the input and $W_Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$. Each query vector represents "what information am I looking for?"
- **Keys (K)**: Generated by $K = XW_K$ where $W_K \in \mathbb{R}^{d_{\text{model}} \times d_k}$. Keys act as indexed labels for each position's content.
- **Values (V)**: Generated by $V = XW_V$ where $W_V \in \mathbb{R}^{d_{\text{model}} \times d_v}$. Values contain the actual information to be retrieved.

The attention computation proceeds in four steps:

1. **Similarity Computation**: QK^T produces a $n \times n$ matrix where entry (i, j) represents how much position i should attend to position j. The dot product naturally measures similarity between query and key vectors.

2. **Scaling**: Division by $\sqrt{d_k}$ (where d_k is the dimensionality of the keys) prevents the dot products from becoming too large. Without scaling, large dot products push the softmax function into regions with extremely small gradients. For example, if $d_k = 512$, dot products could reach magnitudes of ± 20 or more, causing softmax to output distributions close to one-hot vectors.

3. **Normalization**: The softmax function converts similarity scores into a probability distribution: $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$. This ensures attention weights sum to 1 and creates a differentiable selection mechanism.

4. **Weighted Aggregation**: The attention weights are applied to the value vectors, producing a weighted sum that represents the attended information for each position.

2.2.3 Multi-Head Attention Architecture

The Transformer employs multi-head attention so that each head can focus on different parts of the input. Some heads might track syntax, others semantics, positions, etc. This is done by projecting the input matrices Q, K, V by h times using different learned linear projections:

$$(QW_i^Q, KW_i^K, VW_i^V) \tag{2.2}$$

Given this, each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

As such, the final equation for multi-head attention is:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.4}$$

Each head uses different projection matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, where typically $d_k = d_v = d_{\text{model}}/h$ for h heads.

Multi-head attention essentially gives the model multiple different perspective of the same input, and it is a vital system that contributes to make this architecture so powerful.

2.2.4 Positional Information Integration

Since attention is permutation-invariant, the Transformer requires explicit positional information. The original work uses sinusoidal positional encodings:

$$PE_{(\mathrm{pos},2i)} = \sin(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}}) \tag{2.5}$$

$$PE_{(\mathrm{pos},2i+1)} = \cos(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}}) \tag{2.6}$$

These encodings have useful properties: they create unique representations for each position, allow the model to learn relative positions through linear combinations, and can theoretically handle sequences longer than those seen during training.

This architecture enables the Transformer to model dependencies between any pair of positions with constant path length, while maintaining full parallelizability during training.

2.2.5 Encoder-Decoder Transformers

Transformer-based models generate text by predicting the next token in a sequence, conditioned either on an external input (encoder-decoder models) or on the sequence generated so far (decoder-only models). The former is what the original Transformer architecture [15] follows.

- The **encoder** processes the full input sequence in parallel, producing a series of contextual embeddings that capture the meaning and structure of the input tokens.
- The **decoder** generates the output sequence token by token, using the embeddings from the encoder.

This design enables the model to generate outputs that are grounded in the input sequence, rather than generating new text from scratch. As such, it is well-suited for sequence-to-sequence tasks such as machine translation.

2.2.6 Decoder-Only Transformers

In decoder-only Transformers, there is no separate encoder module. The model predicts each token based only on the previously generated tokens, modeling the joint distribution $p(x_1, x_2, \ldots, x_n)$ autoregressively. This means that the model outputs one token at a time, appends it to the input, and repeats until a stopping condition is met.

These models are simpler and more scalable, and they are the ones employed in large language models such as GPT [10], and of course LLaMA [21].

Decoder-only architectures are ideal for open-ended text generation and other tasks where the output is not conditioned on a separate input sequence. Figure 2.2 shows a comparison of encoder-decoder and decoder-only transformer architectures.

2.3 The LLaMA family of models

The *Large Language Model Meta AI* (LLaMA) family [23], developed by Meta AI, comprises a series of transformer-based autoregressive language models designed as competitors to OpenAI's GPT model family. The original LLaMA models were introduced in early 2023, followed by LLaMA 2 [22] later that year and LLaMA 3 in 2024.

The models vary in the number of parameters and capabilities, with the newer versions offering improved performance over the older ones. In particular, with LLaMA 3, Meta introduced models at 8B and 70B parameters with major upgrades in both training scale and performance over LLaMA 2, while approaching results to OpenAI's GPT-4 [11] [23].

The main feature of LLaMA, however, is its open-weight release strategy. Unlike most proprietary models, Meta provides access to the model weights under a research-friendly license, enabling the broader research community and industry practitioners to experiment with, fine-tune, and deploy large-scale language models without relying on closed systems. This openness has led to a proliferation of derivative models, and of course the making of the project described in this document.

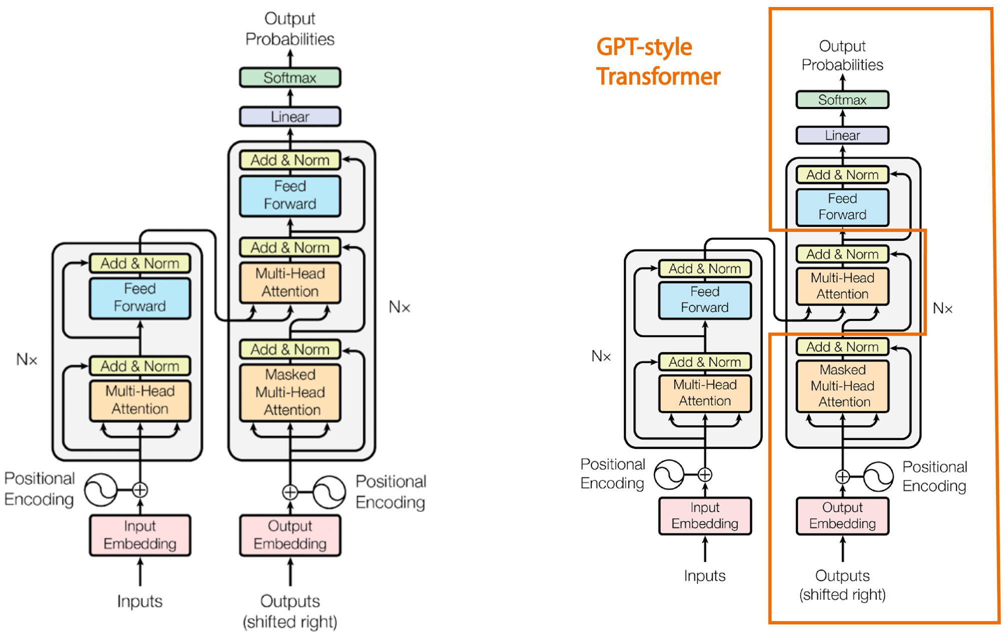


Figure 2.2: On the left, a standard Transformer architecture, which consists of an encoder and a decoder. On the right, we can observe the same architecture, while highlighting the only the layers used by a GPT-style model, which is a decoder-only transformer. The main difference is that the decoder does not attend to the encoder's output, allowing for auto-regressive generation.

2.4 A look at the target hardware

2.5 Relevant compression and optimization techniques

speak about distillation, paper about width+depth pruning.

Chapter 3

Methodology and Implementation

Intro here

3.1 Preliminary research: Franken-LLaMA

3.2 The starting point

speak about llama1b, distillation and why is it hard.

3.3 Pruning methods

3.3.1 Depth-wise pruning

3.3.2 Width-wise pruning (WANDA-based)

3.3.3 LoRA

3.4 Quantization and EoRA

3.5 Evaluation criteria

Chapter 4

Experimental Results and Analysis

nada

4.1 Implementation of the Pipeline

4.1.1 Depth-wise Pruning

4.1.2 Width-wise Pruning (WANDA-based)

ù

4.1.3 LoRA

4.1.4 Quantization and EoRA

4.2 Evaluation

How were the models evaluated?

Chapter 5

Conclusion and Future Work

intro here

5.1 Preliminary observation of the results

Before examining the results based on In this section, we will present and comment on some examples of text generated by the

5.2 Results on TriviaQA

5.3 Results on WikiText

Bibliography

[1] Mike Perkins, *Academic Integrity considerations of AI Large Language Models in the post-pandemic era: ChatGPT and beyond*, https://www.researchgate.net/publication/368775737_Academic_integrity_considerations_of_AI_Large_Language_Models_in_the_post-pandemic_era_ChatGPT_and_beyond.

[2] Daniel Mügge, *AI Is Threatening More Than Just Creative Jobs—It's Undermining Our Humanity*, https://www.socialeurope.eu/ai-is-threatening-more-than-just-creative-jobs-its-undermining-our-humanity.

[3] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, Jeff Dean, *Carbon Emissions and Large Neural Network Training*, https://arxiv.org/abs/2104.10350.

[4] Thomas Spencer, Siddharth Singh, *What the data centre and AI boom could mean for the energy sector*, https://www.iea.org/commentaries/what-the-data-centre-and-ai-boom-could-mean-for-the-energy-sector

[5] Nidhal Jegham, Marwen Abdelatti, Lassad Elmoubarki, Abdeltawab Hendawi *How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference* https://arxiv.org/pdf/2505.09598v1

[6] Google, *Google Environmental Report 2023*, https://sustainability.google/reports/google-2023-environmental-report-executive-summary/

[7] Pengfei Li, Jianyi Yang, Mohammad A. Islam, Shaolei Ren, *Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models*, `https://arxiv.org/abs/2304.03271`.

[8] Arpan Suravi Prasad, Moritz Scherer, Francesco Conti, Davide Rossi, Alfio Di Mauro, Manuel Eggimann, Jorge Tómas Gómez, Ziyun Li, Syed Shakib Sarwar, Zhao Wang, Barbara De Salvo, Luca Benini *Siracusa: A 16 nm Heterogenous RISC-V SoC for Extended Reality with At-MRAM Neural Engine*, `https://arxiv.org/abs/2312.14750`.

[9] Llama Team, AI @ Meta, *Llama 3 models*, `https://www.llama.com/models/llama-3/`

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, *Improving Language Understanding by Generative Pre-Training*, `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf`.

[11] OpenAI, *GPT-4 is OpenAI's most advanced system, producing safer and more useful responses*, `https://openai.com/index/gpt-4/`.

[12] Chris Nicholson, *A Beginner's Guide to LSTMs and Recurrent Neural Networks*. `https://skymind.ai/wiki/lstm`.

[13] S. Hochreiter, J. Schmidhuber, *Long Short-Term Memory*, `doi:10.1162/neco.1997.9.8.1735`.

[14] Mustafa Abbas Hussein Hussein, Serkan Savaş *LSTM-Based Text Generation: A Study on Historical Datasets*, `https://arxiv.org/abs/2403.07087`.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, *Attention is All You Need*, `https://arxiv.org/abs/1706.03762`.

[16] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, `https://arxiv.org/abs/1409.0473`.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, `https://arxiv.org/abs/1810.04805`.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, `https://arxiv.org/abs/1912.01703`.

[19] Angelo Galavotti, *FRANKEN-LLAMA code repository*, `https://github.com/AngeloGalav/franken-llama`

[20] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, Yejin Choi *HellaSwag: Can a Machine Really Finish Your Sentence?* `https://arxiv.org/abs/1905.07830`.

[21] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, et al., *LLaMA: Open and Efficient Foundation Language Models*, `https://arxiv.org/abs/2302.13971`.

[22] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, et al., *Llama 2: Open Foundation and Fine-Tuned Chat Models*, `https://arxiv.org/abs/2307.09288`.

[23] Llama Team, AI @ Meta, *The Llama 3 Herd of Models*, `https://arxiv.org/abs/2407.21783`.

[24] Hanjuan Huang, Hao-Jia Song, Hsing-Kuo Pao, *Large Language Model Pruning*, `https://arxiv.org/abs/2406.00030`.

[25] Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, Hyoung-Kyu Song, *Shortened LLaMA: Depth Pruning for Large Language Models with Comparison of Retraining Methods*, https://arxiv.org/abs/2402.02834.

[26] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, Danqi Chen *Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning*, https://arxiv.org/abs/2310.06694.

[27] ModelCloud, *GPTQModel*, https://github.com/ModelCloud/GPTQModel.

[28] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, Dan Alistarh *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*, https://arxiv.org/abs/2210.17323

[29] Mingjie Sun, Zhuang Liu, Anna Bair, J. Zico Kolter, *A Simple and Effective Pruning Approach for Large Language Models*, https://arxiv.org/abs/2306.11695.

[30] Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, et al., *EoRA: Fine-tuning-free Compensation for Compressed LLM with Eigenspace Low-Rank Approximation*, https://arxiv.org/abs/2410.21271.

[31] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, *LoRA: Low-Rank Adaptation of Large Language Models*, https://arxiv.org/abs/2106.09685.

[32] Mandar Joshi, Eunsol Choi, Daniel S. Weld, Luke Zettlemoyer, *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension*, https://arxiv.org/abs/1705.03551.

[33] Stephen Merity, Caiming Xiong, James Bradbury, Richard Socher, *Pointer Sentinel Mixture Models*, https://arxiv.org/abs/1609.07843.

[34] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, `https://arxiv.org/pdf/1910.10683`.

[35] Chen Liang, Haoming Jiang, Zheng Li, Xianfeng Tang, Bin Yin, Tuo Zhao, *HomoDistil: Homotopic Task-Agnostic Distillation of Pre-trained Transformers*, `https://arxiv.org/abs/2302.09632`.

Acknowledgements

Pending