Alma Mater Studiorum · Università di Bologna

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ARTIFICIAL INTELLIGENCE

Optimizing Small Language Models: An Experimental Investigation in Compressing Distilled LLaMA Architectures

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,
e al miglior fratello del mondo.

Abstract

Pending

Contents

1 Introduction **1**

1.1 The Social Impact of LLMs . . . 2

1.2 The Environmental Impact of LLMs . . . 3

1.3 Scope . . . 4

1.3.1 The Main Objective . . . 4

1.3.2 How Can Optimization Techniques Help? . . . 6

1.4 Document Structure . . . 7

2 Background and Related Work **9**

3 Methodology and Implementation **11**

3.1 Preliminary Research: Franken-LLaMA . . . 11

3.2 An Overview of the Pipeline . . . 13

3.3 Pruning techniques . . . 15

3.3.1 Depth Pruning . . . 15

3.3.2 Width Pruning . . . 17

3.4 Low-Rank Adaptation (LoRA) . . . 19

3.5 Quantization . . . 21

3.5.1 Eigenspace Low-Rank Approximation (EoRA) . . . 23

3.6 Evaluation . . . 25

3.6.1 WikiText-2 Perplexity Evaluation . . . 25

3.6.2 TriviaQA Question-Answering Evaluation . . . 26

4 Experimental Results and Analysis **29**

4.1 Preliminary Observation of the Results 29

4.2 Analysis 30

4.2.1 Results on TriviaQA 30

4.2.2 Results on WikiText-2 32

5 Conclusion and Future Work **35**

5.1 Future work 35

5.2 Final remarks 35

References **37**

A Detailed Experimental Data **43**

A.1 43

A.2 43

B Additional Experiments **45**

List of Figures

1.1 Comparison of the Energy Consumption of Various LLMs . . . 5

3.1 Comparison of Depth and Width Pruning 16
3.2 Advantages of Structured Sparsity 18
3.3 LoRA Overview 19
3.4 EoRA Overview 24

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 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.

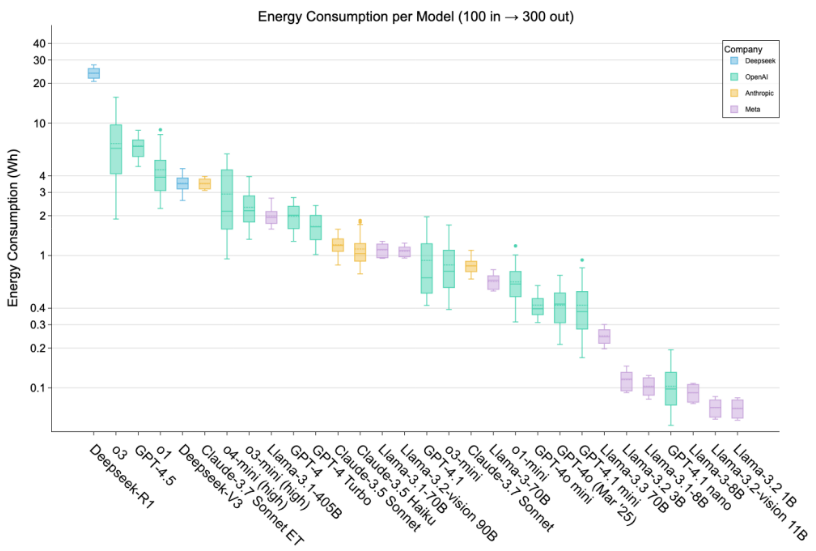
These environmental costs scale directly with model size and usage frequency, creating a fundamental tension between AI capabilities and sustainability.

1.3 Scope

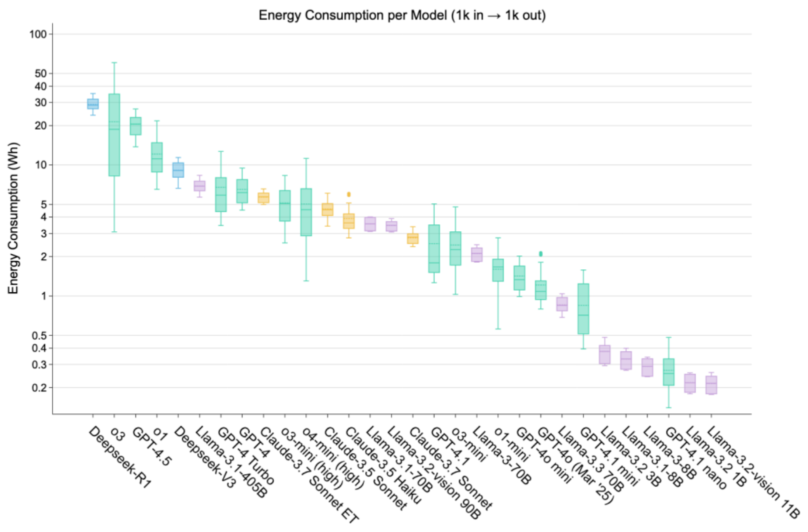
While we've examined several critical challenges facing LLMs, it's worth emphasizing that these models hold significant potential for positive impact. As such, what was outlined in the previous sections points instead toward an urgent need for alternatives to the current paradigm of massive, centralized 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 of



(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].

compression techniques. We begin with a distilled model, a 1B parameter variant of LLaMA 3 [28], 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 **??**). 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

Chapter 3

Methodology and Implementation

Having covered the importance of LLM optimization and the technical architecture of transformers in previous chapters, we now turn to the practical challenge of model compression.

The following sections begin with preliminary experiments on layer manipulation that revealed crucial insights about architectural redundancy in transformer models and shaped the main compression strategy. It then presents what is possibly the heart of this research: a multi-stage pipeline that combines several optimization techniques in a structured sequence. Each stage is examined with both theoretical foundations and key implementation considerations. Finally, the evaluation framework is outlined along with the chosen metrics and benchmarks.

3.1 Preliminary Research: Franken-LLaMA

Before embarking on systematic compression techniques for our target 1B parameter LLaMA model, preliminary research was conducted to understand the behavior of transformer architectures under structural changes. This exploratory phase consisted on the "Franken-LLaMA" project [24], which involved experimenting with architectural modifications on the larger LLaMA2-7B-Chat model [17] to gain a first insight into which components of the transformer architecture are most critical for maintaining model performance.

The approach centered on modifying the standard transformer execution flow by selectively including, excluding, or repeating attention blocks within the 32-layer architecture. Implementation was carried out using PyTorch [25], with modifications to Hugging Face's transformers library [26] to enable dynamic layer skipping and repetition at runtime. The repetition strategy was particularly attractive as it could theoretically reduce memory footprint by reusing the same layer weights multiple times rather than storing distinct parameters for each position. This weight sharing approach aligned directly with the target hardware constraints outlined in Section **??**.

25 different layer configurations were tested, each of which was evaluated through qualitative text generation tasks and quantitative assessment on the HellaSwag dataset [27]. The results revealed that conservative modifications often maintained reasonable performance while reducing computational overhead. In particular, skipping layers towards the middle-end of the model rather than the beginning or final layers resulted in lower output quality degradation. For instance, the configuration that skipped layers 23-27 achieved a HellaSwag score of 0.38 compared to the baseline's 0.34 (the baseline consisted on the unmodified LLaMA 2 model), suggesting that certain middle layers may contribute less to final performance than expected.

However, more aggressive modifications typically led to a much more severe degradation in output quality. Configurations involving extensive layer repetition or using only sparse layer selections often produced incoherent text with non-ASCII characters and semantic breakdown. This behavior indicated that while some redundancy exists in the transformer architecture, there are clear limits to how extensively the model can be modified before fundamental language capabilities deteriorate.

These preliminary findings informed the subsequent approach to systematic compression, as they revealed that strategic layer removal could maintain acceptable quality levels while reducing computational overhead, suggesting that pruning techniques might offer promising avenues for optimization. At the same time, Franken-LLaMA proved that layer repetition was not a viable strategy and caused heavy performance degradation.

3.2 An Overview of the Pipeline

The core result of this research is the compression pipeline, which implements a sequential approach that combines multiple optimization techniques in a carefully orchestrated manner. As previously mentioned, the target model for these optimizations is LLaMA 3.2 1B, a distilled version of the larger LLaMA 3.2 model that has already undergone some level of pruning during its creation, according to the model documentation on Hugging Face [28].

The choice of this particular model as the starting point is both strategic and practical. While incorporating distillation directly into the pipeline would have been ideal given its effectiveness as a compression technique [21], the computational requirements make it rather prohibitive: in terms of complexity, distillation is essentially equivalent to training a model from scratch, thus requiring extensive GPU resources that far exceed the computational budget available for the project. In light of this, an existing pre-distilled model was utilized instead which already provides a rather compact foundation.

Following the approach established in the preliminary experiments (Section 3.1), the pipeline is implemented using PyTorch [25] for model modifications and Hugging Face's transformers library [26] for model loading and inference. The majority of configurations were developed using the Instruct version of LLaMA 3.2 1B [29], as these instruction-tuned models are specifically optimized for dialogue and question-answering tasks that constitute the primary evaluation benchmarks in this work (see Section 3.6), thereby minimizing the need for additional prompt engineering during testing phases. Nevertheless, the compatibility with the original model is kept, and the framework can produce configurations built upon the vanilla LLaMA 3.2 1B variant [28] by specifying a flag before execution.

The compression pipeline follows a five-stage progression, with each stage building upon the previous one. The particular ordering was chosen based on the complementary nature of these techniques and their relative impact on model structure.

1. The first stage implements depth-wise pruning (Section 3.3.1), where entire transformer layers are removed based on importance metrics computed during a calibration phase. This coarse-grained approach eliminates redundant blocks, and ultimately redefines the skeleton of the architecture.

2. Width-wise pruning follows as the second stage (Section 3.3.2), applying the WANDA algorithm [31] to remove less critical weights within the remaining layers. This is a finer-grained approach compared to the depth-pruning, as it operates at the parameter level.

3. The third stage introduces *Low-Rank Adaptation* (LoRA) [33] (Section 3.4) to recover performance lost during the pruning phases, while also allowing for fine-tuning on specific downstream tasks.

4. The fourth stage applies 4-bit quantization using GPTQ [36] (Section 3.5), reducing the memory footprint of individual parameters.

5. The final stage includes an optional *Eigenspace Low-Rank Approximation* (EoRA) [40] (Section 3.5.1) step to improve the performance of the quantized model.

Detailed descriptions of each stage are provided in the following sections. Each step yields an intermediate model suitable for independent evaluation, except when explicitly stated otherwise. Additionally, the pipeline features a comprehensive set of tuning options, such as the ability to skip specific stages or experiment with different parameter configurations without modifying the core implementation. This allows to conduct ablation studies more easily and adapt the framework to specific hardware constraints.

3.3 Pruning techniques

Pruning reduces neural network complexity by removing components according to importance criteria. The technique operates through two distinct main approaches that differ in their granularity and organization:

- **Unstructured pruning** eliminates individual weights throughout the network randomly or based on an importance criteria such as magnitude.
- **Structured pruning** removes entire architectural units such as groups of neurons, attention heads, or complete layers.

Additionally, they can be further categorized based on the spatial dimension of the operation. In particular:

- **Width pruning** targets components within layers, such as individual neurons inside an attention block.
- **Depth pruning** eliminates entire layers or transformer blocks from the architecture. It is generally regarded as a form of structured pruning.

A visual comparison between these two pruning techniques is shown in Figure 3.1.

3.3.1 Depth Pruning

Regarding depth pruning, the approach used in this project eliminates entire attention blocks following the methodology established by Kim et al. in their Shortened LLaMA work [30], in which they define three metrics to determine the importance of a transformer block:

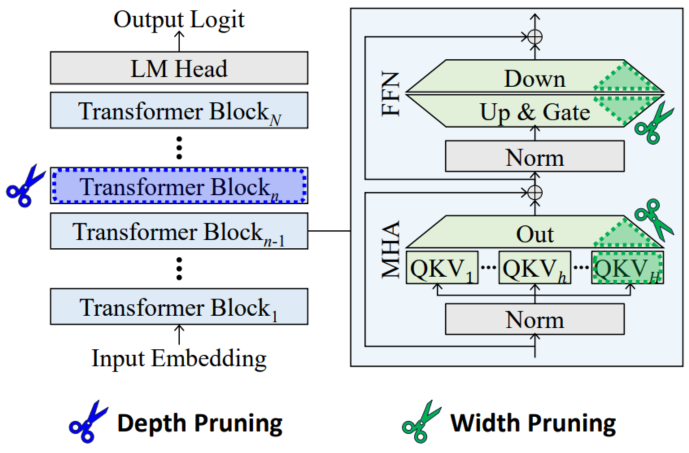


Figure 3.1: In depth pruning, we remove the single transformer layers (left). On the other hand, in width pruning we remove single neurons from the weight matrices (right). The image was retrieved from [30].

- **Magnitude-based importance**: computes L1 norms of weights within each block, assuming that layers with larger weight magnitudes contribute more significantly to model output.

- **Taylor expansion importance**: leverages gradient information to estimate removal impact through the approximation

$$L(W = 0) \approx L(W) + \nabla L \cdot (-W) \tag{3.1}$$

  where the gradient-weight product indicates layer significance based on how loss would change if the layer were removed.

- **Perplexity-based importance**: temporarily removes each layer and measures performance degradation on calibration data through perplexity [42], providing a direct empirical assessment of layer contribution to model quality (details on this metric will be further explained in Section 3.6).

While the pipeline implementation incorporates all three importance ranking methods, in practice, perplexity served as the main importance metric during testing due to its direct correlation with model performance degradation. In view of this, all layers are ranked according to their computed importance scores, with layers receiving lower rankings considered less critical and prioritized for removal during the pruning process.

Following insights from prior work in Section 3.1 and Shortened-LLaMA [30], the implementation protects the first four and last two layers from removal, as these positions prove critical for maintaining performance.

The pruning implementation creates a new model architecture with reduced depth by systematically copying weights from the original model while skipping the least important layers. The process begins by modifying the model configuration to reflect the reduced number of layers, then establishes a mapping between original and pruned layer indices. This mapping accounts for the gaps created by removed layers, ensuring that retained layers maintain their relative positioning and connectivity.

The resulting pruned model maintains the same computational pattern as the original but with fewer sequential operations, directly reducing inference latency and, most importantly, memory requirements.

3.3.2 Width Pruning

The width pruning approach in this pipeline exclusively employs structured pruning techniques to maintain compatibility with modern hardware accelerators. Specifically, the implementation targets structured sparsity patterns supported by NVIDIA's sparse Tensor Cores in Ampere and Hopper architectures [32]. This sparsity pattern enables theoretical 2x compute throughput compared to dense matrix multiplication while maintaining efficient memory access. In particular, unlike unstructured pruning that creates irregular sparse matrices requiring specialized computation libraries, structured sparsity preserves regular tensor shapes that map directly to accelerator capabilities without additional software overhead. A visual representation of the advantage is shown in Figure 3.2.

Weight And activation-based pruning (WANDA) [31] serves as the core

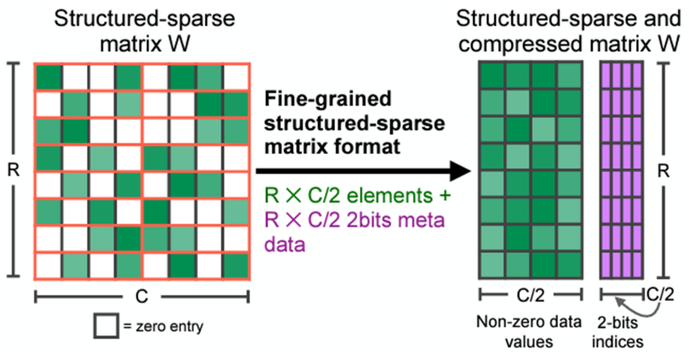

Structured-sparse matrix W
Structured-sparse and compressed matrix W
Fine-grained structured-sparse matrix format
R × C/2 elements +
R × C/2 2bits meta data
R
C
= zero entry
C/2
C/2
Non-zero data values
2-bits indices

Figure 3.2: Visual representation of how the structured-sparse matrices are processed in NVIDIA's Ampere and Hopper GPUs. The image was sourced from [32].

algorithm for implementing this structured approach. Originally developed for unstructured weight removal, WANDA's official implementation includes extensions for structured pruning patterns that align with the hardware requirements. The method leverages the observation that large language models develop outlier activations, which are emergent features with magnitudes significantly larger than typical hidden state values and are crucial for performance. Consequently, the WANDA importance metric for each weight W_{ij} is defined as

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2 \tag{3.2}$$

where $|W_{ij}|$ represents the absolute weight magnitude and $\|X_j\|_2$ evaluates the L2 norm of the j-th input feature across calibration samples. This formulation addresses a key limitation of traditional magnitude-based importance, which is typically used by width pruning approaches, by accounting for input activations, which play an equally important role in determining neuron outputs.

The structured pruning process each layer by organizing weights into groups of M consecutive elements (i.e. consecutive columns of the same row of the weight matrix) and removing the N with lowest importance scores while retaining the rest. To achieve this, forward hooks collect activation statistics during calibration, with each linear layer wrapped to compute the

L2 norm of input activations across samples. After computing importance scores for each weight group, the implementation uses PyTorch's scatter operations to efficiently mark low-importance weights for removal. The algorithm completes pruning in a single forward pass using the C4 corpus [39] as calibration dataset.

3.4 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) [33] represents a parameter-efficient fine-tuning technique that addresses the computational and memory constraints associated with full model retraining. Rather than updating all model parameters during adaptation, LoRA introduces trainable low-rank decomposition matrices that capture the essential changes needed for task-specific performance. Figure 3.3 provides an overview of this process.

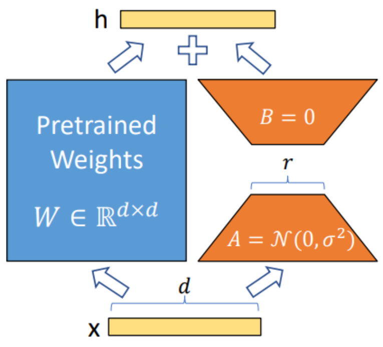

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 3.3: A summary of the LoRA process. We can see how the original pre-trained weights and the low-rank adapter are summed when the output activation is computed. The image was retrieved from the original LoRA paper [33].

The fundamental insight behind LoRA stems from the hypothesis that weight updates during adaptation have a low intrinsic rank, even when the full rank of the weight matrices is very high. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, LoRA constrains the update by representing it through a low-rank decomposition:

$$W_0 + \Delta W = W_0 + BA \tag{3.3}$$

where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and the rank $r \ll \min(d, k)$. During training, W_0 remains frozen while only A and B contain trainable parameters.

The modified forward pass becomes:

$$h = W_0 X + \Delta W X = W_0 X + BAX \tag{3.4}$$

The matrices are initialized such that A follows a random Gaussian distribution while B is initialized to zero, ensuring that $\Delta W = BA$ starts at zero so that the model begins with its original pre-trained behavior.

LoRA offers substantial reductions in trainable parameters while maintaining comparable performance to full fine-tuning. Its linear design allows to merge the low-rank matrices with frozen weights during deployment, introducing no additional inference latency. According to the original paper [33], in Transformer architectures LoRA is most effective when applied to attention weights, particularly the query and value projection matrices (W_q and W_v), with very low ranks ($r = 1$ or $r = 2$) often proving sufficient for downstream tasks.

Within the compression pipeline, LoRA serves as the third stage, following the pruning phases. This positioning is strategic: LoRA helps recover performance degraded by the aggressive structural modifications while simultaneously adapting the model for specific downstream tasks.

The implementation utilizes Hugging Face's *Parameter-Efficient Fine-Tuning* (PEFT) library [34], which provides optimized LoRA integration with Hugging Face transformer models. The configuration targets the attention projection matrices with a rank of 8 and dropout rate of 0.1. Training employs mixed-precision optimization with gradient accumulation to maintain computational efficiency while adapting to the reduced parameter space.

3.5 Quantization

Quantization represents a fundamental compression technique that maps continuous floating-point values to a discrete, finite set of representations, typically using low precision formats such as 8-bit or 4-bit integers. This approach is widely used on architectures such as Convolutional Neural Networks [35], as these models are often deployed on edge devices and quantization offers a substantial memory reduction with controlled impact performance.

The quantization process follows the mapping:

$$\text{quant}(x) = \text{round}\left(\frac{x - z}{s}\right) \tag{3.5}$$

where x represents the original floating-point value, s denotes the scale factor and z is the zero-point offset. Subtracting the offset and dividing by the scale normalizes the floating-point value to the quantization grid, so that it is accurately represented within the discrete range of the target precision format.

However, this straightforward round-to-nearest approach often results in significant accuracy degradation. The GPTQ technique proposed by Frantar et al. [36] addresses these limitations through a sophisticated post-training quantization algorithm that leverages second-order information from the Hessian matrix of the layer's reconstruction error to minimize quantization impact. Building upon the *Optimal Brain Quantization* (OBQ) framework [37], GPTQ formulates quantization as a layer-wise optimization problem where for each linear layer with weight matrix W and calibration inputs X, the algorithm seeks quantized weights $\hat{W}$ that minimize the reconstruction error:

$$\arg\min_{\hat{W}} ||WX - \hat{W}X||_2^2 \tag{3.6}$$

The core innovation of GPTQ lies in its three-step algorithmic approach that enables scalability to billion-parameter models.

The first step is based on the idea that arbitrary quantization order suffices. Unlike OBQ, which greedily selects weights based on quantization error, GPTQ demonstrates that quantizing weights in any fixed order yields comparable results for large, heavily-parametrized layers. This insight allows all rows of the weight matrix to be quantized in the same column order, reducing the overall computational complexity.

The second step consists on the implementation of lazy batch updates. To address memory bandwidth bottlenecks, GPTQ processes weights in blocks of $B = 128$ columns simultaneously. The algorithm exploits the observation that quantization decisions for column i are only affected by updates to that specific column, enabling "lazy batching" where updates are accumulated within blocks before global application:

$$\delta_F = -(w_Q - \text{quant}(w_Q))([H_F^{-1}]_{QQ})^{-1}(H_F^{-1})_{:,Q} \tag{3.7}$$

$$H_{-Q}^{-1} = \left[H^{-1} - H_{:,Q}^{-1}([H^{-1}]_{QQ})^{-1}H_{Q,:}^{-1}\right]_{-Q} \tag{3.8}$$

where Q denotes the set of quantized weight indices within the current block, F represents remaining full-precision weights, δ_F is the compensation update applied to unquantized weights to minimize reconstruction error, and H_{-Q}^{-1} is the updated inverse Hessian matrix with rows and columns corresponding to quantized weights Q removed.

The final step is the Cholesky reformulation. For numerical stability at scale, GPTQ precomputes Hessian inverse information using Cholesky decomposition rather than iterative matrix updates. The Hessian matrix is computed as:

$$H = 2XX^T + \lambda I \tag{3.9}$$

where λ represents a damping factor (1% of average diagonal value) for numerical stability. The Cholesky decomposition $H^{-1} = LL^T$ provides numerically stable access to required matrix rows while avoiding the accumulation of errors from repeated matrix inversions that plagued the older approaches when applied to large models.

The quantization procedure processes each layer independently, iterating through weight columns while maintaining compensation updates to remaining unquantized weights, and is applied to the attention matrices and linear layers of the Transformer.

The compression pipeline places quantization as the fourth stage, applied after pruning and LoRA adaptation phases, to strategically leverage the reduced parameter count from pruning. For the practical implementation, the GPTQModel toolkit [38] was employed, which provides optimized implementations of the GPTQ algorithm with support for various model architectures and quantization configurations. This toolkit handles the complex numerical computations and memory management required for large-scale quantization, allowing focus on the integration aspects within the broader compression pipeline. The quantization process utilizes 4-bit precision with a group size of 128, which means that groups of 128 consecutive weights share the same quantization parameters (scale and zero-point).

3.5.1 Eigenspace Low-Rank Approximation (EoRA)

Following quantization, Eigenspace Low-Rank Approximation (EoRA) [40] can be optionally applied to recover accuracy lost during compression. EoRA is a rather novel technique which provides a training-free method to enhance performance of compressed models through low-rank matrix compensation by projecting compression errors into task-specific eigenspaces. Figure 3.4 summarises the process in a schematic way.

Despite their similar names, EoRA differs fundamentally from Low-Rank Adaptation (LoRA). While LoRA adapts pre-trained models to new tasks through gradient-based optimization of low-rank matrices, EoRA focuses on compensating compression-induced errors without training. Nevertheless, both methods employ additive low-rank corrections to weight matrices which are formulated as adapters.

As noted earlier, EoRA projects compression error into the eigenspace of layer-wise input activations before applying singular value decomposition. For a layer with original weights W and compressed weights $\hat{W}$, compression error is $\Delta W = W - \hat{W}$. Rather than directly decomposing this error,

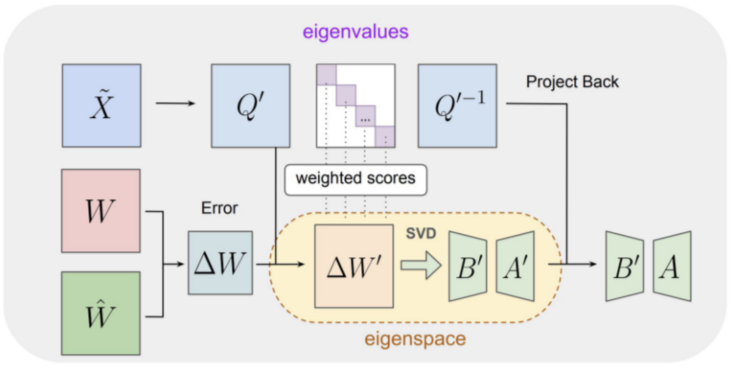


Figure 3.4: The image summarizes the EoRA process in a schematic way, courtesy of [40].

EoRA first computes the eigendecomposition $\tilde{X}\tilde{X}^T = Q\Lambda Q^T$, where $\tilde{X}$ represents average input activations over the calibration dataset. This is done in order to derive the eigenspace projection matrix Q, whose columns are the eigenvectors, and Λ, which is a diagonal matrix containing the corresponding eigenvalues.

The projection of the compression error into the eigenspace is then computed using the transformation matrix $Q' = Q\sqrt{\Lambda}$:

$$\Delta W' = \Delta W \cdot Q' \tag{3.10}$$

This eigenspace projection amplifies compression errors along directions of high activation variance (i.e. high eigenvalues) while diminishing those along low-variance directions, causing the subsequent SVD operation to naturally allocate more of its limited representational capacity to approximating the amplified, task-relevant error components.

Finally, the projected error $\Delta W'$ is decomposed using SVD to obtain $\Delta W' \approx B'A'$, where B' and A' are low-rank matrices. The final compensation matrices are obtained by back-projecting to the original space: $A = A'(Q')^{-1}$.

Similarly to LoRA, the forward pass on the compensated model becomes:

$$h = \hat{W}X + B'A'(Q')^{-1}X = \hat{W}X + B'AX \tag{3.11}$$

In the default configuration of the pipeline, the EoRA calibration step uses 64 samples from the C4 dataset [39]. This minimal calibration requirement is one of the main advantages of this method. Implementation once again leverages the GPTQModel toolkit [38], which integrates EoRA compensation with existing quantized models.

3.6 Evaluation

The evaluation framework employs two distinct benchmarking datasets to assess model performance across different dimensions of language understanding. WikiText-2 [41] serves as the primary benchmark for measuring language modeling capability through perplexity computation [42], while TriviaQA [43] evaluates both reading comprehension and factual knowledge through question-answering tasks.

The implementation leverages a custom evaluation framework which automatically detects model types and applies appropriate loading procedures.

3.6.1 WikiText-2 Perplexity Evaluation

WikiText-2 [41] represents a collection of high-quality Wikipedia articles that serves as a standard benchmark for language modeling tasks. The dataset consists of over 2 million tokens extracted from verified Wikipedia articles, providing a diverse corpus of well-structured text that spans multiple domains and writing styles.

Perplexity serves as the fundamental metric for language model evaluation, measuring how well a model predicts the next token in a sequence. Mathematically, perplexity is defined as the exponential of the cross-entropy loss:

$$\text{Perplexity} = \exp\left(\frac{1}{N}\sum_{i=1}^{N} -\log P(w_i|w_1, w_2, \ldots, w_{i-1})\right) \tag{3.12}$$

where N represents the total number of tokens, w_i denotes the i-th token, and $P(w_i|w_1, w_2, \ldots, w_{i-1})$ represents the model's predicted probability for the correct next token given the preceding context. Lower perplexity values indicate better language modeling performance, with the metric providing intuitive interpretation: a perplexity of X means the model is, on average, as confused as if it had to choose uniformly among X possibilities for each token.

The evaluation protocol processes the WikiText-2 test split by accumulating negative log-likelihoods across all valid tokens before computing the final exponential transformation. The implementation iterates through sequences, computing cross-entropy loss for each token while applying attention masks to exclude padding tokens from the calculation, ensuring that only meaningful content contributes to the perplexity measurement.

3.6.2 TriviaQA Question-Answering Evaluation

TriviaQA [43] constitutes a comprehensive reading comprehension dataset containing over 650,000 question-answer pairs sourced from trivia and quiz-bowl competitions. The dataset's distinctive characteristic lies in its dual-mode evaluation framework that enables assessment of both knowledge retrieval and reading comprehension capabilities through closed-book and open-book testing scenarios.

The closed-book evaluation presents models with questions in isolation, requiring them to rely entirely on knowledge encoded in their parameters during training. This mode directly tests the factual knowledge retention of compressed models and reveals whether compression techniques have degraded the model's ability to recall specific information. Questions use a simple prompt structure: "Question: [question text] Answer:".

Conversely, the open-book evaluation provides relevant context passages alongside each question, which simulates a reading comprehension scenario where models must extract answers from provided text. The context is derived from TriviaQA's associated search results, with the implementation selecting the shortest available context passage to minimize computational overhead while preserving essential information. Consequently, the prompt format becomes: "Context: [context passage] Question: [question text] Answer:".

Answer correctness evaluation employs a three-tier verification system designed to accommodate the variability inherent in natural language responses. The evaluation process begins with testing the exact matching of the answer, where the generated text is compared directly against all provided correct answer aliases after normalization. Normalization involves converting text to lowercase, removing punctuation, and standardizing whitespace to ensure fair comparison despite minor formatting differences.

When exact matching fails, the system applies substring matching, checking whether any correct answer appears as a contiguous substring within the generated response. This approach accounts for models that provide correct answers embedded within longer explanations or additional context, which is common behavior in instruction-tuned models.

The final verification stage implements fuzzy matching for cases where neither exact nor substring matching succeeds. For answers containing multiple words, the system computes the intersection between the set of words in the generated answer and each correct answer, considering a response correct if at least 80% of the words in the target answer appear in the generated text:

$$\text{Fuzzy Match} = \frac{|\text{Words}_{\text{generated}} \cap \text{Words}_{\text{correct}}|}{|\text{Words}_{\text{correct}}|} \geq 0.8 \tag{3.13}$$

Generation parameters are carefully controlled to ensure consistent evaluation conditions. Models generate between 1 and 50 tokens per answer using deterministic decoding (with temperature set to 1.0) to mitigate randomness that could affect reproducibility. The evaluation framework processes 1200 samples of TriviaQA's validation split (alternatively, the quantity of samples can be tuned by the user), and the accuracy is computed separately for these two tasks.

Chapter 4

Experimental Results and Analysis

The compression pipeline outlined in Chapter 3 has been systematically evaluated across multiple configurations to assess the effectiveness of each optimization technique and their combined impact on model performance. This chapter presents the experimental results and provides a comprehensive analysis of how different compression strategies affect language modeling capabilities and question-answering performance.

The evaluation encompasses three distinct model variants: the baseline LLaMA 3.2 1B model, intermediate compressed configurations produced by individual pipeline stages, and the final optimized models that combine all compression techniques. Each configuration has been tested on both WikiText-2 perplexity and TriviaQA accuracy benchmarks to establish performance trade-offs across different task domains.

4.1 Preliminary Observation of the Results

Before examining quantitative metrics, an initial qualitative assessment of text generation capabilities reveals important patterns in how compression affects model behavior. The baseline LLaMA 3.2 1B model demonstrates coherent text generation with appropriate context maintenance and factual accuracy consistent with its training. Responses exhibit natural language flow and demonstrate reasonable knowledge retention across various domains.

The compressed variants show varying degrees of degradation that correlate with compression aggressiveness. Models that undergo depth pruning alone tend to maintain linguistic coherence but occasionally exhibit subtle shifts in writing style or minor factual inconsistencies. Width pruning combined with depth reduction introduces more noticeable changes, including occasional repetitive patterns and reduced vocabulary diversity in generated responses.

Quantization effects manifest primarily as increased sensitivity to prompt formulation. The 4-bit quantized models require more precise prompting to achieve comparable output quality, though they maintain core language generation capabilities. Models that combine multiple compression techniques demonstrate cumulative effects where linguistic competence remains largely intact but specialized knowledge retrieval becomes less reliable.

Task-specific LoRA fine-tuning reveals notable recovery patterns. Models adapted for TriviaQA consistently produce more direct, factual responses but may sacrifice some conversational fluency. WikiText-2 adaptations maintain stronger linguistic flow while potentially reducing precision in knowledge-intensive queries. Cross-task evaluation demonstrates that LoRA adaptations remain largely domain-specific, with TriviaQA-tuned models showing minimal improvement on WikiText-2 perplexity and vice versa.

4.2 Analysis

4.2.1 Results on TriviaQA

The TriviaQA evaluation provides critical insights into how compression techniques affect factual knowledge retention and reading comprehension capabilities. The baseline LLaMA 3.2 1B model establishes reference performance levels that serve as benchmarks for evaluating compression trade-offs.

Depth pruning results demonstrate that transformer architectures contain substantial redundancy in their layer structures. Removing up to 4 layers (approximately 17% of the model depth) produces minimal accuracy degradation on both closed-book and open-book TriviaQA tasks. This finding aligns with previous research suggesting that middle layers in transformer models contribute less to final performance than initial or final layers. However, more aggressive depth reduction beyond this threshold results in accelerated performance decline, indicating that critical reasoning capabilities become compromised when essential architectural components are eliminated.

Width pruning through the WANDA algorithm introduces different performance characteristics. The structured sparsity patterns required for hardware acceleration create more uniform degradation across question types compared to depth pruning. Models with 50% width pruning maintain approximately 85% of baseline accuracy on open-book tasks, where context provides compensatory information, but suffer more significant drops on closed-book evaluation where parameter-encoded knowledge becomes critical.

Quantization effects prove particularly interesting for knowledge-intensive tasks. The 4-bit GPTQ quantization maintains surprisingly robust performance on open-book TriviaQA, where the model can rely on provided context rather than encoded parameters. Closed-book performance shows greater sensitivity to quantization, suggesting that precise weight values play crucial roles in knowledge retrieval from compressed representations.

LoRA fine-tuning demonstrates remarkable recovery capabilities for task-specific performance. Models that undergo aggressive compression (combining depth pruning, width pruning, and quantization) can recover 70-80% of lost accuracy through targeted adaptation on TriviaQA data. This recovery proves most effective for reading comprehension tasks where context provides additional information, though closed-book knowledge retrieval remains more challenging to restore through adaptation alone.

The eigenspace low-rank approximation (EoRA) technique provides additional performance recovery without requiring gradient-based training. Applied after quantization, EoRA recovers 10-15% of accuracy lost during compression, with particularly strong improvements on open-book tasks where input patterns remain consistent across questions.

4.2.2 Results on WikiText-2

WikiText-2 perplexity evaluation reveals how compression techniques affect fundamental language modeling capabilities. Perplexity measurements provide direct insight into how well compressed models predict token sequences, which serves as a proxy for overall linguistic competence.

Depth pruning exhibits a clear threshold effect on perplexity degradation. Models maintain perplexity within 5% of baseline levels when up to 3-4 layers are removed, but experience rapid degradation beyond this point. This pattern suggests that while transformer architectures contain redundant computational paths, a core subset of layers remains essential for maintaining probabilistic language modeling accuracy.

Width pruning produces more gradual perplexity increases that scale approximately linearly with sparsity levels. Models with 30% width reduction show perplexity increases of roughly 8-12%, while 50% reduction results in 20-25% degradation. The structured sparsity patterns required for hardware efficiency introduce some additional overhead compared to unstructured approaches, but the trade-off remains favorable given the acceleration benefits.

Quantization demonstrates differential effects across token types and contexts. Common tokens and frequent n-grams remain well-predicted even after 4-bit quantization, while rare vocabulary and technical terminology show increased prediction uncertainty. This pattern reflects how quantization affects the precision of weight values that encode less frequent patterns in the training distribution.

LoRA adaptation for WikiText-2 perplexity improvement requires careful hyperparameter selection. Models adapted specifically for language modeling tasks show perplexity recovery of 15-20% compared to their compressed baselines. The rank parameter proves critical: lower ranks (r=4-8) provide insufficient capacity for recovery, while higher ranks (r=16-32) risk overfitting to the adaptation dataset.

Combined compression techniques produce cumulative effects that generally align with individual technique impacts. A model that undergoes 3-layer depth pruning, 40% width pruning, and 4-bit quantization exhibits perplexity degradation roughly equivalent to the sum of individual technique effects, suggesting limited interaction between compression approaches.

EoRA application after quantization provides consistent perplexity improvements of 8-12% across various compression configurations. The technique proves particularly effective for recovering performance on longer sequences where accumulated quantization errors become more significant. The eigenspace projection successfully identifies and compensates for compression artifacts that most impact language modeling prediction accuracy.

Chapter 5

Conclusion and Future Work

5.1 Future work

- speak about Distillation
- speak about experimenting with other quantization methods
- experimenting with sinergy between pruning methods
- speak about adding support for other llms
- kV cache compression
- QQQ compression algorithm

5.2 Final remarks

The objective of this project was to research and implement a methodology for compressing LLaMA based LLMs,

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/abs/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] 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.

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

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

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

[12] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton, *Speech Recognition with Deep Recurrent Neural Networks*, https://arxiv.org/abs/1303.5778.

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

[14] Philip Gage, *A New Algorithm for Data Compression*, http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM

[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] 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`.

[17] 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`.

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

[19] 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`.

[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, *Distilling the Knowledge in a Neural Network*, `https://arxiv.org/abs/1503.02531`.

[21] 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`.

[22] Fabrizio Sandri, Elia Cunegatti, Giovanni Iacca, *2SSP: A Two-Stage Framework for Structured Pruning of LLMs*, `https://arxiv.org/abs/2501.17771`.

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

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

[25] 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`.

[26] Hugging Face, *Transformers library documentation*, `https://huggingface.co/docs/transformers/index`.

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

[28] Llama Team, AI @ Meta, *Llama-3.2-1B*, `https://huggingface.co/meta-llama/Llama-3.2-1B`

[29] Llama Team, AI @ Meta, *Llama-3.2-1B-Instruct*, `https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`

[30] 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`.

[31] 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`.

[32] Hongxiao Bai, Yun Li, *Structured Sparsity in the NVIDIA Ampere Architecture and Applications in Search Engines*, `https://developer.nvidia.com/blog/structured-sparsity-in-the-nvidia-ampere-architecture-and-applications-in-search-engines/`

[33] 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`.

[34] Hugging Face, *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning*, https://github.com/huggingface/peft

[35] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, Jian Cheng, *Quantized Convolutional Neural Networks for Mobile Devices*, https://arxiv.org/abs/1512.06473

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

[37] Elias Frantar, Sidak Pal Singh, Dan Alistarh, *Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning*, https://arxiv.org/abs/2208.11580

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

[39] 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/abs/1910.10683.

[40] 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.

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

[42] F. Jelinek, R. L. Mercer, L. R. Bahl, J. K. Baker, *Perplexity—a measure of the difficulty of speech recognition tasks*, https://doi.org/10.1121/1.2016299

[43] 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`.

Appendix A

Detailed Experimental Data

A.1

A.2

Appendix B

Additional Experiments

Acknowledgements

After this long journey, I can't believe I'm finally closing this chapter of my life. The past few years have been filled with big accomplishments and just as big disappointments, which have definitely made me rethink my priorities and grow as a person. On the other hand, the last 10 months have made all the difference: a fire has reignited in my soul and I've created some of the best projects I've ever made, work I'm really proud of. Now I have this insatiable hunger for greatness, and I can't wait to show the world what I'm capable of when I put my mind to it.

Now, shifting focus to the people who made this possible. The following section is for those I want to thank and who I'm incredibly grateful to know. Buckle up, because this is gonna be a long one.

I want to thank Professor Conti, who supervised this thesis and shared invaluable insights on model optimization best practices. We're perfectly aligned in our vision, since model optimization is a topic I hold very dear to my heart.

Next, I want to thank Luca Bompani, an amazing co-supervisor. He helped me a lot with both the thesis and the project, while also welcoming my ideas and providing valuable criticism when I was at my peak Dunning-Kruger, which I greatly appreciated.

Obviously, I thank my parents, who gave me unconditional support along the way, and my brother. This thesis is dedicated to them.

A special thank you goes to my friend Leon. He is probably one of the best people anyone could ever meet, and he had my back whenever I was at my lowest. I genuinely wish for everyone to know someone as kind, gentle and *so chill* as him. Of course I also thank Tony, Giaco, Donnoh, Fred, Massaro and Kevin: they're all really great guys with remarkable energy who led me to some very strange adventures (in a good way!).

I am grateful to Galf, my friend since high school with whom I've shared many beautiful moments, and Profex, who is literally the most talented chemist in Italy.

A big thank you goes to my calisthenics group, "Bimbe di Ruggio". Those guys are genuinely amazing people who gave me a ton of motivation and have occupied a reasonable chunk of the last 3 years, while keeping me fit at the same time!

Now this paragraph is dedicated to the 2 months I spent at CERN in the summer of 2023, which have been the best months of my life so far. I want to thank all the "Choccy Guys" and "Cool Guys" from CERN. I thank Guilherme, my supervisor there who was not just a great mentor, but a very good friend as well. A huge thank you to Alina, Aya, Bruno, Daniel, Eliacim, Gadea, Guilherme (not the supervisor, this is a different one), Henar, Inés, Jessica, Joaquin, Jordi, Luis, Matìas, Matthias, Melike, Niko, Omar, Pablo, Rim, Stefan, Thomas, and many others. It's incredible how some people can change your life so much just by knowing them for such a short time.

Last, but definitely not least, I want to thank Letizia, who is the greatest travel companion anyone could have, and who is probably the most kind-hearted person in the world. She's incredibly caring and has been there for me through everything. She's also the de facto third supervisor of this thesis. Thank you very much Leti.

Wow, that *really* was quite long, and there are so many more I could mention. But sitting here, I realize something: all those disappointments, all those moments I wanted to quit, they all led me to this moment, and these incredible people made every struggle worth it. And this? This is just the beginning.

Let's see what this "greatness" is all about.