Leandro Giusti Mugnaini

Streamlining Giants: Towards Efficient Compression of Large Language Models

São Paulo

2026

Leandro Giusti Mugnaini

Streamlining Giants: Towards Efficient Compression of Large Language Models

Revised Version

Dissertation presented to the Graduate Program in Electrical Engineering at Escola Politécnica da Universidade de São Paulo in partial fulfillment of the requirements for the degree of Master of Science.

Concentration Area: Computer Engineering

Advisor: Prof. Dr. Artur Jordão Lima Correia

São Paulo

2026

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: gov.br Documento assinado digitalmente LEANDRO GIUSTI MUGNAINI Data: 08/09/2026 13:23:26-0300 Verifique em https://validar.iti.gov.br

Assinatura do orientador: gov.br Documento assinado digitalmente ARTUR JORDAO LIMA CORREIA Data: 07/09/2026 16:45:20-0300 Verifique em https://validar.iti.gov.br

Ficha Catalográfica

Mugnaini, Leandro
Streamlining Giants: Towards Efficient Compression of Large Language Models / L. Mugnaini -- versão corr. -- São Paulo, 2026.
63 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.INTELIGÊNCIA ARTIFICIAL 2.APRENDIZADO COMPUTACIONAL 3.APRENDIZAGEM PROFUNDA 4.REDES NEURAIS I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Elaborada pela Divisão de Biblioteca da Escola Politécnica da USP, por meio de formulário eletrônico, com os dados fornecidos pelo(a) autor(a)

Acknowledgements

I would like to dedicate this work to all the people who, in different ways, helped me accomplish something of which I am very proud.

First of all, I would like to thank my parents, Eden and Isabel, for always believing in me, no matter how difficult the challenges I had to face were. I dedicate this work to you both!

I would also like to thank Laisa, Leonardo, Bruno, and Alyson for supporting me throughout this process, encouraging me to take the research further, and helping me not to give up during difficult times. To all my research colleagues, especially Gustavo, Bruno, Lucas, Victor, Keith, and Carolina: thank you for all the meetings, conversations, insights, and hard work. I feel privileged to have known you, and I am certain that I shared this journey with brilliant people who are building bright futures.

My sincere thanks also go to Prof. Anna Reali and the C2D team for all their support, for believing in my work, and for trusting my skills. I was privileged to have been given many opportunities that I will never forget.

I am especially grateful to my supervisor, Prof. Artur Jordão, for accepting me as his student, for guiding me through this path, and for changing my life in ways I still cannot fully describe. I leave this program with a different mindset, new opportunities, and a much better understanding of what science truly is.

Finally, I would like to thank Instituto de Ciência e Tecnologia Itaú (ICTi) and Programa Unificado de Bolsas Itaú (PBI). This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2023/11163-0. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. I would like to thank grant #402734/2023-8, National Council for Scientific and Technological Development (CNPq).

Resumo

Mugnaini, L.G. **Streamlining Giants: Towards Efficient Compression of Large Language Models.** 2026. Dissertação (Mestrado em Engenharia de Computação) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2026.

O Aprendizado Profundo impulsiona uma nova onda em sistemas computacionais e desencadeia a automação de problemas cada vez mais complexos. Em particular, os Grandes Modelos de Linguagem (*Large Language Models - LLMs*) avançaram significativamente nas tarefas cognitivas, frequentemente igualando ou até superando o desempenho humano. No entanto, seu elevado número de parâmetros resulta em altos custos computacionais e lento processo de inferência, tornando desafiadora a implantação em ambientes com recursos limitados. Entre as estratégias para superar tal desafio, a poda destaca-se como um mecanismo eficaz, pois reduz o tamanho do modelo enquanto mantém sua capacidade preditiva. Neste trabalho, propomos uma nova estratégia de poda para enfrentar os problemas mencionados. Nosso método comprime LLMs de forma eficiente, removendo estruturas menos críticas na *Multi-Head Attention* e *Multilayer Perceptron*. Para avaliar a importância das estruturas internas, projetamos os dados de entrada sobre os pesos, medindo sua magnitude total. Ao eliminar os componentes com as menores importâncias, superamos as limitações das técnicas existentes, que muitas vezes carecem de flexibilidade ou eficiência. Em particular, nosso critério supera o atual estado da arte em tarefas de raciocínio de senso comum, alcançando uma alta taxa de compressão com impacto mínimo no desempenho preditivo (*zero-shot*). Especificamente, nosso método supera outras técnicas em até 9,52 pontos percentuais de acurácia, ao mesmo tempo em que proporciona uma inferência até 1.25× mais rápida, sem exigir hardware especializado. Além disso, nossa abordagem reduz o consumo de energia em até 20%, contribuindo diretamente para a diminuição da pegada de carbono em implantações de LLMs. Confirmamos a flexibilidade de nossa técnica em diferentes famílias de LLMs, incluindo LLaMA e Phi.

Palavras-chave: Grandes modelos de linguagem, Poda, Poda estruturada, Compressão de modelos, Inteligência artificial verde

Abstract

Mugnaini, L.G. **Streamlining Giants: Towards Efficient Compression of Large Language Models.** 2026. Dissertation (Master's in Computer Engineering) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2026.

Deep learning drives a new wave in computing systems and triggers the automation of increasingly complex problems. In particular, Large Language Models (LLMs) have significantly advanced cognitive tasks, often matching or even surpassing human-level performance. However, their extensive parameters result in high computational costs and slow inference, posing challenges for deployment in resource-limited settings. Among the strategies to overcome the aforementioned challenges, pruning emerges as a successful mechanism since it reduces model size while maintaining predictive ability. In this work, we propose a novel pruning strategy to overcome the aforementioned problems. Our technique efficiently compresses LLMs by removing less critical structures within Multi-Head Attention (MHA) and Multilayer Perceptron (MLP). To score the importance of internal structures, we project the input data onto weights, measuring the overall magnitude. By removing the components with the lowest scores, we overcome the limitations of existing approaches, which often fall short in flexibility or efficiency. In particular, our criterion surpasses the current state-of-the-art (SOTA) on commonsense reasoning tasks, achieving a high pruning ratio with minimal impact on predictive ability. Specifically, our method outperforms other SOTA techniques by up to 9.52 percentage points in accuracy while delivering up to 1.25× faster inference without requiring specialized hardware. Furthermore, our approach reduces energy consumption by up to 20%, directly contributing to the reduction of the carbon footprint in LLM deployments. We confirm the flexibility of our technique on different families of LLMs, including LLaMA and Phi.

Keywords: LLM, Pruning, Structured pruning, Model compression, Green AI

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	Artificial Intelligence
AMP	Attention Heads and MLP Pruning
ARC-c	AI2 Reasoning Challenge – Challenge
ARC-e	AI2 Reasoning Challenge – Easy
BIG-Bench	Beyond the Imitation Game Benchmark
BPE	Byte-Pair Encoding
COPAL	Continual Pruning in Large Language Generative Models
CR	Column-Row
FLOPS	Floating-Point Operations per Second
GLUE	General Language Understanding Evaluation
GPQA	Google-Proof Question Answering
GPT	Generative Pre-Trained Transformer
GPU	Graphics Processing Unit
GQA	Grouped-Query Attention
GRU	Gated Recurrent Unit
KV	Key-Value
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short-Term Memory
LVLM	Large Vision–Language Model
MHA	Multi-Head Attention
MLP	Multilayer Perceptron
MMLU	Massive Multi-Task Language Understanding

MNIST	Modified National Institute of Standards and Technology dataset
MQA	Multi-Query Attention
NLP	Natural Language Processing
OBS	Optimal Brain Surgeon
PEFT	Parameter-Efficient Fine-Tuning
PIQA	Physical Interaction Question Answering
PPL	Perplexity
ReLU	Rectified Linear Unit
RIA	Relative Importance and Activation
RMSNorm	Root Mean Square Normalization
RNN	Recurrent Neural Network
RoPE	Rotary Position Embedding
SDPA	Scaled Dot-Product Attention
SiLU	Sigmoid Linear Unit
SOTA	State-of-the-art
SQuAD	Stanford Question Answering Dataset
SVD	Singular Value Decomposition
SwiGLU	Swish-Gated Linear Unit
VRAM	Video Random Access Memory

Contents

1 Introduction

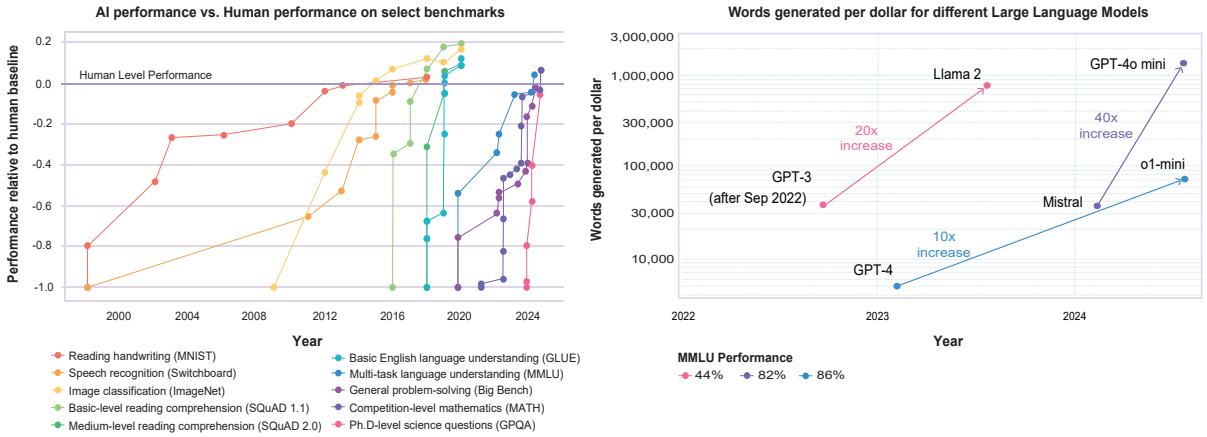
Within the evolving landscape of Artificial Intelligence (AI), LLMs stand out as a pivotal force, propelling Natural Language Processing towards unprecedented stages – often matching or even surpassing human-level performance in many language benchmarks (OPENAI, 2023; ZHOU et al., 2023; BROWN et al., 2020). Remarkably, Figure 1 (Left) confirms that models once trailing human performance now rapidly surpass it within months and establish new state-of-the-art results.

In terms of computational costs, LLMs are also more efficient over time. Figure 1 (Right) shows a concrete example: models that previously delivered only 30,000 words per dollar now exceed one million words in a single year of improvement. However, despite these gains in efficiency and performance [2], state-of-the-art LLMs continue to scale in size.

[1] Available at: https://www.gov.uk/government/publications/international-ai-safety-report-2025/international-ai-safety-report-2025. Accessed: 18 May 2025.

[2] Unless otherwise stated, we define efficiency in terms of computational demand, so that higher efficiency corresponds to lower computational cost. We define performance as the general accuracy across benchmarks.

Figure 1 – Left: AI benchmark scores relative to human performance from 1998 to 2024. Over two decades, models have steadily closed the human-machine gap across multiple tasks. Remarkably, on the most challenging benchmarks, such as MATH and GPQA, AI relative performance jumped from roughly -1 to around 0 (Human-Level) in a very short period of time. We also highlight that earlier results relied on specialized machine-learning models rather than general-purpose models – a characteristic of LLMs. **Right:** Evolution of cost-efficiency – in words generated per US dollar (log scale) – for general-purpose LLMs with comparable MMLU performance. Since 2022, these models have delivered dramatically more words per dollar while improving accuracy. Efficiency rose as much as 40×, from approximately 30,000 words per dollar at 44% accuracy (GPT-3) to over 1 million words per dollar at 82% accuracy (GPT-4o mini).



Source: Adapted from International AI Safety Report 2025[1].

Models, such as DeepSeek-V3 (DEEPSEEK-AI, 2024) and LLaMA-3.1 (GRATTAFIORI et al., 2024), now reach the impressive mark of hundreds of billions of parameters, increasing their computational and latency demands. Additionally, training an 8 billion parameter model produces up to 420 tons of equivalent CO_2 – the same as 83 years of energy consumption for the average home in the United States of America (MORRISON et al., 2025). Furthermore, the largest model providers are producing hundreds of billions of tokens per day, rapidly recouping their training costs (MORRISON et al., 2025). These factors drive the development of more sustainable models and focus the scientific community on the environmental impact of AI, an emerging field known as Green AI (SCHWARTZ et al., 2020).

The trade-off between model capacity and computational demands drives the development of strategies capable of reducing resource consumption without sacrificing predictive ability. Among existing strategies, we highlight knowledge distillation (ZHOU et al., 2022; CHEN; ZHAO, 2019; SAEEDI et al., 2022), quantization (XIAO et al., 2023; HU et al., 2021; FRANTAR et al., 2023; JIN et al., 2024), and pruning (MA et al., 2023; SUN et al., 2024). Recent studies confirm pruning as a promising solution for compressing models, as it maintains predictive ability and is often hardware-agnostic (GAO et al., 2024b; ASHKBOOS et al., 2024; OUDERAA et al., 2024). The central idea of this technique involves removing the least important components from the model. Pruning strategies span various granularities, from removing fine-grained elements (e.g., weights, neurons) to coarse-grained structures (e.g. filters, attention heads, layers).

Within the field of LLMs, pruning techniques fall into three main categories: structured, semi-structured and unstructured pruning (WAN et al., 2023; CHENG et al., 2024). Structured pruning removes entire components – such as attention heads or layers – while preserving the overall network structure (ZHU et al., 2024). Semi-structured (a.k.a. structured $N : M$) pruning promotes model sparsity by removing groups of consecutive parameters following a pruning mask (FRANTAR; ALISTARH, 2023). Finally, unstructured pruning removes individual weights without considering any pattern, increasing model sparsity by zeroing out parameters (FRANTAR; ALISTARH, 2023; SUN et al., 2024).

Regardless of the category, pruning relies on importance metrics to guide its decisions. For example, magnitude-based methods use the absolute value of weights to determine less important structures (FRANTAR; ALISTARH, 2023; KIM et al., 2024). In contrast, a data-driven approach takes into account weights and input data in component evaluation (SUN et al., 2024). Loss-based metrics assess the importance of the structure by measuring their impact on the loss function. Successful methods in this field typically employ Taylor expansion (MA et al., 2023; KIM et al., 2024; WANG et al., 2019). Different techniques also incorporate regularization (WANG et al., 2021; ZHANG

et al., 2022). Finally, some approaches consider the similarity between weights or representations, employing metrics such as cosine similarity to determine the least important structures (GROMOV et al., 2025).

In summary, these importance metrics define how we evaluate and rank model components before removing them. Accurately removing only unimportant or redundant elements prevents representational collapse and low downstream predictive ability. Therefore, selecting and formulating appropriate metrics directly affect the trade-off between model size and task performance, highlighting their critical role in reliable model compression.

1.1 Motivation

Despite the rich variety of pruning strategies described above, according to the AI Index Report (MASLEJ et al., 2025), the development of state-of-the-art LLMs still remains in the hands of a few private companies due to extraordinarily high computational and financial requirements. According to Figure 2, academia alone accounts for no notable models in 2024 [3], while industry organizations produce the overwhelming majority. This concentration creates a significant democratization bottleneck, placing cutting-edge architectures beyond the reach of most academic institutions and small enterprises.

The high computational and financial requirements also amplify the energy and climate stakes of large-scale training and inference. The Global Carbon Budget annual report expects 38.1 billion tonnes of fossil CO_2 emissions in 2025, which marks a 1.1% rise and sets a new record [4]. In parallel, AI workloads push electricity demand upward: in the United States, data centers drive about half of the growth in total electricity demand from now to 2030 [5]. Major technology firms such as Google, Meta, Amazon, and Microsoft also pursue nuclear options, via new builds and reactor restarts, to deal with the energy requirements of large-scale computing [6].

To alleviate the accessibility gap and reduce the carbon footprint of these large-scale architectures, researchers aim to develop compression and acceleration techniques – including parameter pruning – for open-source models.

[3] Epoch AI, an AI Index data provider, uses the term "notable model" to designate particularly influential AI models. The criterion includes state-of-the-art advancements, historical significance, large training cost, with significant use, or high citation rates.

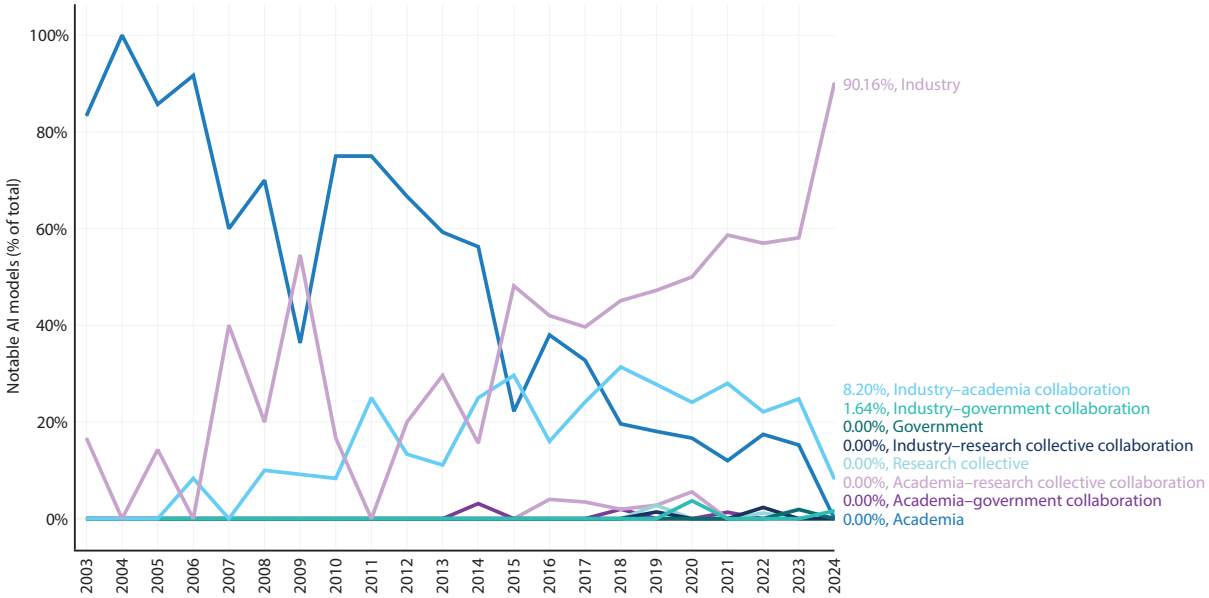
[4] Since 2001 and with the help of the scientific community, the Global Carbon Project establishes a common and mutually agreed knowledge base to support policy debate and action to slow down the increase of greenhouse gases in the atmosphere. Available at: https://globalcarbonbudget.org/. Accessed: 16 December 2025.

[5] Available at: https://www.technologyreview.com/2025/11/20/1128167/future-of-electricity/. Accessed: 16 December 2025.

[6] Available at: https://www.technologyreview.com/2025/05/20/1116339/ai-nuclear-power-energy-reactors/. Accessed: 16 December 2025.

[7] Available at: https://www.gov.uk/government/publications/international-ai-safety-report-2025/international-ai-safety-report-2025. Accessed: 18 May 2025.

Figure 2 – Notable AI models (% of total) by sector, from 2004-2024. Epoch AI, an AI Index data provider, considers a model notable if it is highly cited (over 5000 citations), or has a large training cost (over $1,000,000), or has significant use (over 1 million monthly active users), or achieves state-of-the-art performance, or has an equivalent level of historical significance. Since the mid-2010s, the development of notable AI models has progressively shifted from academia to industry. This transition coincides with the consolidation of deep learning and, later, large-scale Transformer-based architectures, whose training demands substantial computational and financial resources, concentrating state-of-the-art model development in organizations with access to large compute clusters and specialized engineering infrastructure.



Source: Adapted from International AI Safety Report 2025[7].

Despite promising results, existing pruning strategies often face trade-offs involving performance, latency, or the requirement for specialized hardware. It turns out that by removing or zeroing internal components, pruning alters the learned representations of the model, which degrades predictive ability if carried too far. Consequently, there remains a need for simpler, hardware-agnostic techniques that effectively compress LLMs with minimal performance loss while achieving tangible inference speedups.

To directly overcome the aforementioned limitations, we propose a novel pruning technique that addresses the gaps in current methods and reduces these trade-offs. We achieve this by leveraging the projection of input data onto the weights to determine the importance of the internal components in each layer of the LLM.

1.2 Research Statement and Questions

To guide this investigation, we formulate a set of research questions. More specifically, we aim to clarify (i) how to systematically identify and prune the least relevant

structures in LLMs? (ii) What are the trade-offs of pruning on model accuracy, inference latency, energy use, environmental footprint, and associated financial costs? (iii) In what ways does removing different internal components of LLMs influence their predictive ability? (iv) How do architectural differences across LLMs create challenges for designing model-agnostic pruning techniques?

By addressing these questions, we confirm the following research statement:

Projecting input data onto model weights using the ℓ_1-norm enables us to effectively estimate the contribution of internal components in Large Language Models, thereby quantifying their importance to the final representation. Consequently, removing components with the lowest contributions from our importance ranking yields models that are lighter, faster, and maintain their predictive capacity.

1.3 Objectives

From a theoretical standpoint, our goal is to establish a rigorous criterion for identifying and ranking the internal components of LLMs by projecting input data onto weights. We aim to show that this projection reliably quantifies the contribution of each attention head and MLP neuron, allowing the removal of the least critical components.

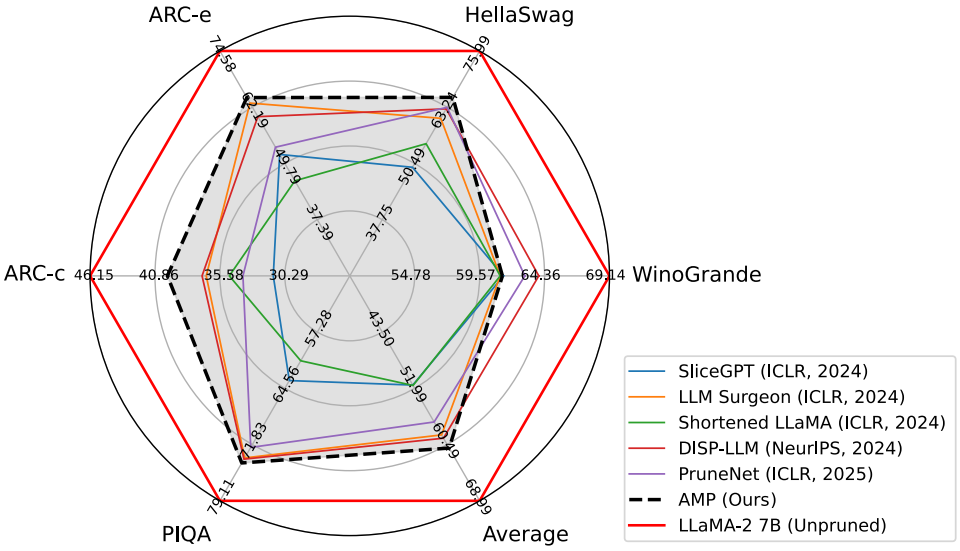
On the practical side, our goal is to translate these concepts into a novel pruning criterion that produces more efficient LLMs. Specifically, we seek a strategy that (i) reduces the computational costs and latency associated with running LLMs, (ii) preserves the predictive ability of pruned models, (iii) is also hardware-agnostic. By providing such benefits, we have the objective to create more sustainable AI models, aiming to minimize environmental impact and making these architectures more accessible to the general public.

1.4 Contributions

The primary contribution of this dissertation lies in the development of a criterion for pruning Large Language Models (LLMs), which culminated in the creation of AMP (Attention Heads and MLP Pruning), a structured pruning method that surpasses state-of-the-art techniques of LLM pruning. More precisely, our contributions are the following:

- Building directly on our research statement, we reveal that projecting input data onto weights successfully discerns critical from non-critical components. We confirm the effectiveness of our strategy with a coherence experiment, an essential validation we believe should become standard in pruning research.
- Our development aligns with Green AI principles when deploying LLMs at scale, leading to direct cost savings and reduced CO_2 emissions. Moreover, our criterion

Figure 3 – Radar chart comparing zero-shot performance of various pruning techniques against the unpruned LLaMA-2 7B (in red) across five benchmarks (WinoGrande, HellaSwag, ARC-e, ARC-c, PIQA) and their overall average. Each colored polygon corresponds to different state-of-the-art (SOTA) pruning techniques. The closer the vertices of a polygon lie to the center, the nearer its predictive ability is to random-guess accuracy; polygons whose outlines reach the outer edge approximate the predictive ability of the original model. We highlight that AMP (dashed black), our pruning technique, surpasses SOTA techniques in most benchmarks and on average results.



Source: Prepared by the author.

identifies unimportant structures within minutes, requiring a negligible fine-tuning process for performance recovery, making it both practical and resource-efficient.

- Compared to existing approaches, our technique achieves up to 1.25× inference speedup without the need for specialized hardware. It also surpasses existing structured pruning techniques by up to 1.49 percentage points in average task accuracy. Figure 3 highlights our results against state-of-the-art techniques.

- We evaluate our criterion in various settings – including standard benchmarks and widely adopted open-source LLMs – and confirm that it generalizes across different families, requiring minimal modifications.

Additionally, to allow the reproducibility of our experiments, we make the source code publicly available at https://github.com/c2d-usp/Efficient-LLMs-with-AMP and the model checkpoints at https://huggingface.co/collections/c2d-usp/efficient-large-language-models. Unlike existing works, we also provide the weights, allowing interested readers to run our models without the need of re-executing the full pipeline.

1.5 Publications

As a result of the aforementioned contributions, we have published the following papers during this research:

- Leandro Giusti Mugnaini, Bruno Lopes Yamamoto, Lucas Lauton de Alcantara, Victor Zacarias, Edson Bollis, Lucas Pellicer, Anna Helena Reali Costa, Artur Jordao. *Efficient LLMs with AMP: Attention Heads and MLP Pruning*. International Joint Conference on Neural Networks, 2025. (MUGNAINI et al., 2025b)
- Leandro Giusti Mugnaini, Carolina Tavares Duarte, Anna Helena Reali Costa, Artur Jordao. *Layer Pruning With Consensus: A Triple-Win Solution*. IEEE Access, 2025. (MUGNAINI et al., 2025a)

Besides the previous publications, our work also contributes to the following studies:

- Carolina Tavares Duarte, Leandro Giusti Mugnaini, Gustavo Henrique do Nascimento, Ian Pons, Keith Ogawa, Guilherme Stern, Lucas Libanio, Aline Paes, Anna Helena Reali Costa, Artur Jordao. *Tiny Titans: Efficient Large Vision, Language and Multimodal Models through Pruning*. Conference on Graphics, Patterns and Images - Tutorial, 2025. (TAVARES et al., 2025)
- Bruno Lopes Yamamoto, Lucas Lauton de Alcantara, Victor Zacarias, Leandro Giusti Mugnaini, Keith Ando Ogawa, Lucas Pellicer, Rosimeire Pereira Costa, Edson Bollis, Anna Helena Reali Costa, Artur Jordao. *Compressing LLMs with MoP: Mixture of Pruners*. ArXiv preprint, 2026. (YAMAMOTO et al., 2026)

1.6 Work Organization

We organize this dissertation as follows. Chapter 1 motivates the problem, states our research questions and contributions, and outlines the structure of the dissertation. Chapter 2 introduces the theoretical foundations of Transformer architectures, the overall pruning process, and evaluation protocols. In Chapter 3, we review the main works related to the contributions of this dissertation. In Chapter 4, we introduce our strategy for pruning Large Language Models. Chapter 5 describes the experimental setup, and shows and discusses the experimental results. In Chapter 6, we present the conclusions of this research and directions for future work.

2 Background

Before the advent of Transformers (VASWANI et al., 2017) – the current foundation for large-scale language models – researchers relied on Recurrent Neural Networks (RNNs) to process sequential data (JÓZEFOWICZ et al., 2015). Unlike traditional Multilayer Perceptron, RNNs implement loops in their connections. At each time step, the network updates a hidden state with the current input, which lets the model retain context throughout the steps. Consequently, RNNs excel at tasks such as time-series forecasting and language modeling (SALINAS et al., 2020; JÓZEFOWICZ et al., 2015).

In practice, however, RNNs gradually overwrite earlier context as they propagate the hidden state, causing information from distant time steps to vanish (PASCANU et al., 2013). Gated architectures, such as Long Short-Term Memory (LSTM) networks (HOCHREITER; SCHMIDHUBER, 1997) and Gated Recurrent Units (GRUs) (CHO et al., 2014), mitigate this effect but still struggle to capture dependencies beyond a few hundred steps (MIKOLOV et al., 2015).

Moreover, RNNs process inputs one time step at a time, preventing parallel computation across sequence positions. This sequential dependency creates significant bottlenecks during both training and inference (VASWANI et al., 2017).

By introducing key innovations such as self-attention and positional encoding, the Transformer overcomes the limitations of RNNs, successfully managing long-range dependencies and enabling parallel processing of input sequences. We begin with a brief background about the Transformer architecture, highlighting the key components necessary to our pruning method.

2.1 Transformers

Transformer architectures fall into three variants: encoder-decoder (initially designed for machine translation), encoder-only (typically used for classification), and decoder-only. We focus our attention on the decoder-only variant (the standard in autoregressive language models).

Internally, the model starts by splitting input text into subword units, called tokens, using a predefined vocabulary – often built via segmentation algorithms, such as byte-pair encoding (BPE) (SENNRICH et al., 2016) or unigram language modeling (KUDO, 2018) – to balance coverage and efficiency. It maps each token to a continuous vector in the embedding matrix. During training, the model refines these vectors so that semantically related tokens lie close in embedding space. Then, it adds fixed, learned, or relative

positional encodings to each vector, giving the model a sense of token order despite the permutation invariance of its attention mechanism. Finally, it combines token embeddings and positional encodings into a single representation for each position and feeds these into the decoder layers for contextual processing.

A stack of decoder layers, each serving as a fundamental computation unit, proceeds with the processing. Each layer works as follows: first, it applies root mean squared layer normalization (RMSNorm) (ZHANG; SENNRICH, 2019) to stabilize activations. Next, the normalized vectors enter the self-attention component, which computes context-aware representations by weighting each token against all others. This component leverages the use of a causal mask so each position only attends to itself and earlier tokens, never to the future, enforcing autoregressive generation. Then, the layer adds the attention result back via a residual connection and applies a second RMSNorm. After normalization, a position-wise multilayer perceptron (MLP) refines each token independently. Finally, the layer merges the MLP output through another residual path.

These multiple steps allow each layer to blend global context (via self-attention) with local non-linear transformations (via the MLP), while preserving gradient flow and numerical stability through normalization and residual connections.

After all decoder layers, most implementations apply one final RMSNorm. Once normalized, the model maps the hidden state of each position into the vocabulary logits. Then, a softmax yields the probability distribution for the next token, completing the autoregressive prediction pipeline.

Since the decoder layers contain the majority of the parameters in Large Language Models, most pruning techniques also remove components from these structures, so we focus our attention on them. We start our deep dive into the concepts of the Transformer by using the original definitions by Vaswani et al. (2017). To provide a more practical view, we highlight some of the most important modifications by famous open-source decoder-only families, such as LLaMA and Phi, from Meta and Microsoft, respectively.

2.1.1 Self-Attention

Self-attention lets each token attend to every other token in a single parallel pass. Consider S the sequence length and d_{model} the hidden size dimension. Given an input $\mathbf{X} \in \mathbb{R}^{\mathrm{S} \times d_{\mathrm{model}}}$, self-attention takes into account Key ($\mathbf{K}$), Query ($\mathbf{Q}$) and Value ($\mathbf{V}$) matrices as follows:

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{K}, \mathbf{Q} = \mathbf{X}\mathbf{W}^{Q}, \mathbf{V} = \mathbf{X}\mathbf{W}^{V}, \tag{2.1}$$

where $\mathbf{W}^{K}, \mathbf{W}^{Q} \in \mathbb{R}^{d_{model} \times d_k}$ and $\mathbf{W}^{V} \in \mathbb{R}^{d_{model} \times d_v}$ are learnable projection matrices, with d_k being the dimensionality of the Key and Query matrices and d_v being the

dimensionality of the Value matrix. Usually, open-source decoder-only models, including LLaMA-2, consider $d_k = d_v$, so for simplicity we denote both by d_k.

Building on these projections, the Scaled Dot-Product Attention (SDPA), a fundamental mechanism in Transformer, computes self-attention in terms of:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}, \tag{2.2}$$

where $\mathbf{M}$ is a causal mask that ensures the model remains autoregressive during generation, by adding $-\infty$ on disallowed (future) positions before softmax.

In SDPA, each Query vector dynamically weights all Value vectors according to their Key similarity, capturing long-range dependencies across the entire sequence in one parallel operation.

2.1.2 Position-Wise Multilayer Perceptron

After attention, a two-layer Multilayer Perceptron (MLP) processes each token $x \in \mathbb{R}^{d_{model}}$ as

$$\text{MLP}(x) = \text{ReLU}(x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.3}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{i}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{i}}}$ project up into an inner dimension d_{i}, and $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{i}} \times d_{\text{model}}}$, $\mathbf{b}_2 \in \mathbb{R}^{d_{\text{model}}}$ project back down.

The LLaMA models re-express this same computation using a gated-activation variant (SwiGLU) and split the weights into three distinct linear projections: Up, Gate, and Down projections. Given an input $x \in \mathbb{R}^{d_{model}}$, the SwiGLU MLP is

$$\text{MLP}(x) = (\text{SiLU}(x\mathbf{W}_{Gate}) \odot (x\mathbf{W}_{Up}))\mathbf{W}_{Down}, \tag{2.4}$$

where $\mathbf{W}_{Up}, \mathbf{W}_{Gate} \in \mathbb{R}^{d_{model} \times d_i}$, $\mathbf{W}_{Down} \in \mathbb{R}^{d_i \times d_{model}}$, and $\odot$ represents an element-wise multiplication.

Similarly to $\mathbf{W}_1$ and $\mathbf{b}_1$ from the original Transformer, the role of $\mathbf{W}_{Up}$ and $\mathbf{W}_{Gate}$ is to project the input x into a higher-dimensional space ($\mathbb{R}^{d_i}$) that enhances representational capacity. The projections are then combined element-wise before being projected back to $\mathbb{R}^{d_{\text{model}}}$ by $\mathbf{W}_{Down}$ (just like $\mathbf{W}_2$ and $\mathbf{b}_2$), allowing residual add, while keeping parameter counts manageable.

Although Up and Down projections reduce overall parameter count, MLP blocks still constitute roughly two-thirds of the total weights in each layer of open-source LLMs such as LLaMA-2 and Phi-2. Consequently, these structures are clear targets for pruning.

2.1.3 Positional Encoding

Unlike RNNs, where each model computes hidden states sequentially, each depending on the previous one, Transformers contain no recurrence or convolution, computing attention across all tokens at once. For the model to take into account the order of the sequence, the model must inject some information about the position of the tokens. Also, for the summation to be possible, the positional encodings must have the same dimension d_{model} as the embeddings. For position pos and index i, we set

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{\text{model}}}), \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{\text{model}}}). \end{aligned} \tag{2.5}$$

Each dimension pair forms a sinusoid whose wavelength grows geometrically from 2π to $10000 \cdot 2\pi$, giving each sequence position a unique wave-pattern vector. Transformer infers both absolute position and relative distance by comparing their phase differences. Vaswani et al. (2017) show that this fixed scheme matches learned positional embeddings without adding parameters.

Modern LLMs typically use an updated version of the proposed positional encoding, called Rotary Position Embedding (RoPE) (SU et al., 2024). RoPE injects positional information directly into the Queries and Keys via a fixed rotation. Each two-dimensional sub-vector of the Query/Key pair rotates by an angle proportional to its token position, allowing the model to encode both absolute positions and relative offsets without learning extra parameters. This continuous, parameter-free scheme preserves rotation equivariance and improves extrapolation to longer sequences.

Since our pruning technique only considers the projection of data onto weights, it is invariant to the positional encoding technique (e.g., absolute, relative, RoPE) and works with a wider variety of models.

2.1.4 Multi-Head Attention

The Transformer splits its d_{model}-dimensional Queries, Keys, and Values into N parallel projections, a mechanism known as Multi-Head Attention (MHA). MHA lets the model attend simultaneously to multiple representation subspaces at each position.

Formally, given an input $\mathbf{X} \in \mathbb{R}^{\text{S} \times d_{\text{model}}}$, each attention head $n \in \{1, ..., N\}$ within MHA performs projections into the Key ($\mathbf{K}$), Query ($\mathbf{Q}$) and Value ($\mathbf{V}$) matrices as follows

$$\mathbf{K}_n = \mathbf{X}\mathbf{W}_n^K, \mathbf{Q}_n = \mathbf{X}\mathbf{W}_n^Q, \mathbf{V}_n = \mathbf{X}\mathbf{W}_n^V, \tag{2.6}$$

where $\mathbf{W}_n^K$, $\mathbf{W}_n^Q$ and $\mathbf{W}_n^V$ are learnable projection matrices belonging to $\mathbb{R}^{d_{model} \times d_k}$ and $d_k = \frac{d_{model}}{N}$ is the dimensionality allocated to each attention head.

Then, for a given decoder-layer, we compute each attention head by using the SDPA (2.2) and express the n-th attention head as

$$\mathbf{h}_n = \text{Attention}(\mathbf{Q}_n, \mathbf{K}_n, \mathbf{V}_n) = \text{Softmax}\left(\frac{\mathbf{Q}_n\mathbf{K}_n^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_n. \tag{2.7}$$

MHA concatenates all N attention heads and multiplies them by the output matrix $\mathbf{W}_O \in \mathbb{R}^{(d_k N)\times d_{model}}$. Specifically,

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N)\mathbf{W}_O. \tag{2.8}$$

The Transformer by Vaswani et al. (2017) employs $N = 8$ and $d_k = 64$, while the LLaMA-2 7B architecture uses $N = 32$ and $d_k = 128$.

2.1.4.1 Key-Value Caching, Grouped-Query Attention and Multi-Query Attention

Although MHA is powerful, its naive implementation becomes expensive at inference time (AINSLIE et al., 2023). During autoregressive decoding, recomputing all past Keys and Values for each new token wastes compute. To avoid this, Key–Value Caching (POPE et al., 2023) retains previous $\mathbf{K}$ and $\mathbf{V}$ matrices. When a new token arrives, the model computes only its own $\mathbf{K}$ and $\mathbf{V}$ and appends them to the cache. This reduces per-token attention cost from $O(S^2)$ to $O(S)$ as the sequence grows.

However, storing and attending over cached tensors still impose significant memory and bandwidth overhead. To mitigate this problem, many models replace standard Multi-Head Attention with Grouped-Query Attention (GQA) (AINSLIE et al., 2023) or Multi-Query Attention (MQA) (SHAZEER, 2019).

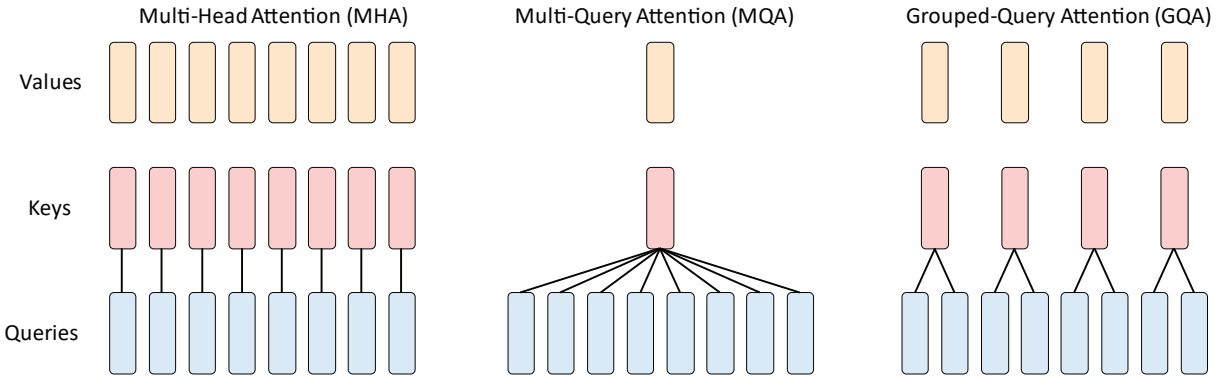
Grouped-Query Attention divides the N heads into G groups. Each group shares Key–Value projections but keeps separate Queries. This cuts cached Key–Value memory by N/G. Popular LLMs such as LLaMA-3 (GRATTAFIORI et al., 2024), Phi-3 (ABDIN et al., 2024), and Mistral-7B (JIANG et al., 2023) adopt GQA. On the other hand, Multi-Query Attention shares one Key and one Value projection across all N Query heads. During autoregressive decoding, it reduces Key–Value memory N-fold. PaLM (CHOWDHERY et al., 2022) corresponds to a popular architecture that uses MQA.

Figure 4 summarizes these architectures in contrast to the traditional MHA.

2.1.4.2 Pruning MHA, GQA, and MQA Attention Mechanisms

Although strategies such as GQA and MQA reduce KV-Cache overhead compared to MHA, the remaining attention heads still impose runtime costs (AINSLIE et al., 2023). Orthogonally to these mechanisms, head pruning surges as a promising technique to accelerate inference and reduce parameter count. Removing redundant or low-impact

Figure 4 – Overview of each attention-based mechanism. Multi-Head Attention has N Query, Key, and Value heads. Multi-Query Attention shares single Key and Value heads across all Query heads. Grouped-Query Attention instead interpolates between MHA and MQA, sharing a single Key and Value heads for each group of Query heads.



Source: Adapted from Ainslie et al. (2023).

heads decreases stored KV entries and dot-product operations, accelerating decoding and the overall inference process.

2.1.5 Root Mean Square Normalization

RMS normalization (ZHANG; SENNRICH, 2019) standardizes each token vector by its root-mean-square norm rather than subtracting a mean. Given input x, RMSNorm computes

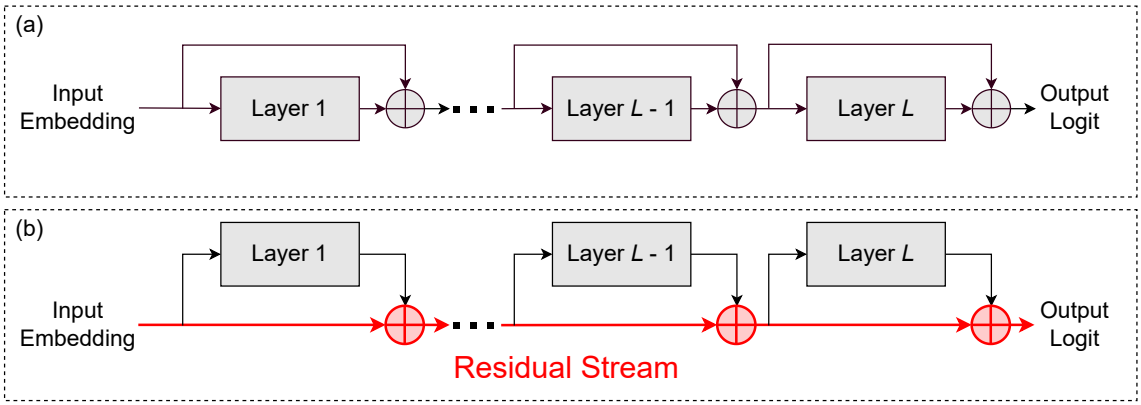
$$y_i = \frac{x_i}{\mathrm{RMS}(x)} * \gamma_i, \quad \text{where} \quad \mathrm{RMS}(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \epsilon}, \tag{2.9}$$

and γ is a learned scale vector. This simpler normalization variant stabilizes activations with lower computational overhead than full LayerNorm (BA et al., 2016), the previous standard normalization method for LLMs.

2.1.6 Residual Stream

Recent advancements in mechanistic interpretability offer novel insights into the internal workings of Transformer-based language models (LINDSEY et al., 2025; ELHAGE et al., 2021; SKEAN et al., 2025). In the decoder layer, both the position-wise Multilayer Perceptron and Multi-Head Attention add their output to their input via residual connections. These shortcuts preserve earlier layer representations, prevent network degradation in deep stacks, and improve gradient flow by providing direct backpropagation paths (HE et al., 2016; VASWANI et al., 2017). Together, they form the residual stream, a critical information pathway in Transformers. Figure 5 contrasts (a) the classic layer-centric view

Figure 5 – Classic vs. Residual Stream Perspectives of a Decoder-Only LLM. In the classic depiction (a), layers lie at the heart of the architecture, and the residual connections are viewed merely as alternative pathways for data flow. In the residual stream perspective (b), a persistent residual stream (in red) becomes the primary data path, with each layer and its internal submodules reading from and writing to it. This view underscores the fundamental role of residual connections in residual neural networks, including LLMs.



Source: Prepared by the author.

with (b) the residual stream perspective, where the persistent residual stream (in red) carries the contributions of every layer of the model.

Elhage et al. (2021) propose interpreting the residual stream of LLMs as a communication channel, where the main blocks of each layer function as independent units that read and write to this stream. As a result, we express the output of the model by aggregating the individual contributions from all internal components.

Building on this perspective, it is possible to measure the relevance of each internal structure by measuring its contribution to the residual stream. If a structure consistently contributes with minor additions to the residual stream, we expect its removal to have minimal impact on the predictive ability of the model. This idea is the core of our pruning criterion, as we detail in Section 4.

2.2 Parameter-Efficient Fine-Tuning (PEFT)

The development of an LLM typically comprises pre-training and post-training stages (GRATTAFIORI et al., 2024). During pre-training, autoregressive language models learn general linguistic and factual patterns from large text corpora through next-token prediction (BROWN et al., 2020; HOFFMANN et al., 2022). Post-training adapts the pre-trained model to specific tasks or behaviours through procedures such as supervised fine-tuning, instruction tuning, alignment, and Parameter-Efficient Fine-Tuning (PEFT) (WEI et al., 2022; OUYANG et al., 2022; HU et al., 2022).

Even after pre-training, adapting LLMs to downstream tasks remains computationally demanding (HOFFMANN et al., 2022). Full fine-tuning typically updates all parameters of a model that may contain billions of weights, requiring large batches, long training schedules, and high-bandwidth GPU memory (BROWN et al., 2020). Parameter-Efficient Fine-Tuning (PEFT) methods address this challenge by freezing the original weights of the backbone and learning a lightweight set of task-specific parameters (DING et al., 2023). Among these techniques, Low-Rank Adaptation (LoRA) stands out for its simplicity and effectiveness (HU et al., 2022). The key assumption is that the update required to specialize a pre-trained weight matrix has low intrinsic rank. Concretely, instead of directly modifying a weight matrix $\mathrm{W}_o \in \mathbb{R}^{d\times k}$, LoRA keeps W_o fixed and learns a low-rank correction $\Delta\mathrm{W}$, parameterized as the product of two smaller matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, with $r \ll min(d, k)$. During fine-tuning, the effective weights become

$$\mathrm{W} = \mathrm{W}_o + \Delta\mathrm{W} = \mathrm{W}_o + AB, \tag{2.10}$$

and the optimization process updates only the entries of A and B. In practice, LoRA inserts these low-rank adapters into a subset of linear projections (e.g., attention and MLP layers), drastically reducing the number of trainable parameters while leaving the forward pass of the frozen backbone unchanged. After training, LoRA either merges the low-rank updates back into Wo or keeps them as separate modules, enabling the modular deployment of multiple task-specific adapters on top of the same base model.

Building on the success of LoRA, several variants refine this idea to better exploit the parameter and memory budgets. Approaches such as QLoRA (DETTMERS et al., 2023) combine low-rank adapters with low-bit quantization of the frozen backbone, allowing fine-tuning of large models using consumer-grade GPUs while keeping most weights in a highly compressed format. Other methods, such as AdaLoRA (ZHANG et al., 2023), allocate higher ranks to layers that appear more task-critical and reduce the rank where simpler updates suffice, trading a fixed global rank for a data-driven distribution of capacity. Collectively, these methods preserve the central principle of PEFT – updating only a small fraction of parameters – while offering flexible design choices to balance memory footprint, and training costs for large-scale LLMs.

Our pruning strategy operates in the post-training stage. The method compresses an existing pre-trained model and subsequently applies PEFT through LoRA to recover part of the predictive ability lost during pruning. This procedure preserves the original pre-training process while reducing the computational requirements of the resulting model.

2.3 Zero, One, and Few-Shot Learning

Providers typically pre-train LLMs on massive text corpora, which enables them to acquire broad linguistic and factual knowledge but leaves them without expertise in specialized downstream tasks (TOUVRON et al., 2023a; TOUVRON et al., 2023b; GRATTAFIORI et al., 2024; DEEPSEEK-AI, 2024). To evaluate how effectively an LLM applies this knowledge to new tasks without extensive retraining, researchers employ three complementary in-context learning paradigms: zero-shot, one-shot, and few-shot.

In zero-shot learning, we present the model with only a natural-language instruction (task description), without any examples, so it must map inputs to outputs using its pre-trained representations. For instance, a zero-shot prompt for sentiment classification might read as follows:

> Classify the sentiment of the following review about this dissertation as *Very Positive* or *Acceptable*.
> Review: "Overall, this dissertation is very good and a pleasure to read."
> Sentiment: _____

In one-shot learning, the practice is to include exactly one input–output exemplar before the target prompt. This exemplar specifies the desired output format, style, and reasoning approach. For example:

> Classify the sentiment of the following review about this dissertation as *Very Positive* or *Acceptable*.
> Example: Review: "Overall, this dissertation is very good and a pleasure to read."
> Sentiment: Very Positive
> Review: "The dissertation is just fine."
> Sentiment: _____

Few-shot learning strikes a balance between lightweight prompting and retraining (GAO et al., 2021). In many tasks, a handful of examples lets models approach – or sometimes match – fine-tuned performance, making few-shot learning a cost-efficient alternative (BROWN et al., 2020). For example:

Classify the sentiment of the following review about this dissertation as *Very Positive* or *Acceptable*.
Example 1: Review: "Overall, this dissertation is very good and a pleasure to read."
Sentiment: Very Positive
Example 2: Review: "The dissertation is just fine."
Sentiment: Acceptable
Example 3: Review: "This dissertation is a solid piece of work. It is clear that considerable effort went into its development, and I enjoyed reading it."
Sentiment: Very Positive
Review: "It is a great work!"
Sentiment: _____

Each paradigm adds task-specific information through the prompt (i.e. the input instruction). Researchers use benchmarks such as BIG-Bench (SRIVASTAVA et al., 2023) and the LM Evaluation Harness (GAO et al., 2024a) to assess how predictive ability scales with model size and pre-training data under zero-, one-, and few-shot settings. Because pruning aims to preserve the predictive ability of the pruned model as close as possible to the original counterpart, many studies evaluate pruned models on zero-shot benchmarks, testing performance based solely on the remaining pre-trained knowledge. For a fair comparison with existing techniques, we follow the same practice.

3 Related Work

The strategy of pruning consists of removing structures from deep networks. Its success in compressing and accelerating deep models is due to two factors: over-parameterization and plasticity of deep models. In the over-parameterized regime, networks have more capacity (i.e. the ability to learn various representations from the data) than required to solve the problem at hand; hence, some structures become redundant and, thus, unimportant (CHANG et al., 2021). Plasticity, on the other hand, refers to the ability of neural networks to recover from damage to their structure (MITTAL et al., 2018). This concept aligns with pruning as it allows for recovering the predictive ability of deep models after removing certain components.

Leveraging the previous factors, Tavares et al. (2025) highlight the three main steps of a typical pruning algorithm. **First**, the algorithm defines the types of components to consider for removal. Strategies typically fall into three main categories: structured, semi-structured, and unstructured. The **second step** establishes a pruning criterion to identify the components for removal. The criterion plays a crucial role, as it defines how pruning decisions occur and directly affects the balance between computational efficiency and predictive capacity. In the **third step**, the algorithm adjusts the weights of the survival structures to compensate for the changes in the architecture and, hence, in predictive ability. This phase ensures the pruned network recovers from structural changes and adapts its parameters to the new architecture.

In computer vision, top-performing pruning algorithms apply the previous process multiple times (SHEN et al., 2022; MURALIDHARAN et al., 2024; GANJDANESH et al., 2024). For example, Pons et al. (2024) propose a similarity-based criterion that identifies unimportant layers by selecting candidates whose internal representations remain closest to those of the dense model, achieving substantial reductions in floating point operations per second (FLOPS) and latency with negligible impact on accuracy. Nascimento et al. (2025) extend this idea to decide, at each pruning step, whether to remove neurons or entire layers, showing that the same criterion guides to different pruning choices over the iterations. By contrast, LLM pruning typically uses a one-shot approach (MA et al., 2023; SUN et al., 2024), also including a single fine-tuning stage. This prevents potential catastrophic forgetting (KOTHA et al., 2024) – the forgetting of previously learned information by the model. Because providers rarely release original training data, and retraining at scale would incur enormous costs, researchers use Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA (HU et al., 2022), and rely on compact datasets such as Alpaca (TAORI et al., 2023).

As the definition of which components to remove is the first fundamental step of any pruning criterion, we contrast our method with other techniques, considering the three main categories of pruning: structured, semi-structured and unstructured. We further discuss the pruning criteria and different fine-tuning strategies to recover some of the lost predictive ability.

3.1 Structured Pruning

LLM-Pruner (MA et al., 2023) identifies and removes tightly coupled neuron clusters by treating each cluster as a discrete unit in the model's computational graph through a structural dependency analysis. This approach ensures that the pruning process maintains the integrity of the Transformer architecture by grouping dependent parameters together. The method calculates an importance score for each group using a first-order Taylor expansion, which estimates the change in the loss function when the system removes each cluster. After pruning the lowest-scoring clusters, LLM-Pruner applies a post-training phase to recover lost performance using LoRA with the Alpaca dataset.

Sheared LLaMA (XIA et al., 2024) prunes an LLM into a compact architecture and then continues pre-training using dynamic batch loading. This technique adjusts the proportion of data from different domains in each training step to prioritize those with higher relative loss, ensuring balanced performance across all data distributions. The method employs a Lagrange-multiplier algorithm to impose shape constraints for a given target architecture. An optimization procedure then minimizes the masked-weight loss to derive pruning masks and updated weights under those constraints. Finally, dynamic batch loading resumes pre-training until the pruned model reaches the loss threshold that a predetermined scaling function on the training set defines.

Unlike the previous methods, Shortened LLaMA (KIM et al., 2024) removes entire Transformer layers based on two importance metrics. First, the Taylor+ criterion measures the significance of each operation in the MLP and MHA modules (e.g., an MHA Query-projection or an MLP Up-projection) by using a first-order approximation of the change in training loss on a calibration dataset when the model omits that operation. The importance of a layer comprises the sum of the importance of its operations. Second, the perplexity-driven criterion evaluates each layer by the change in held-out text perplexity with versus without the layer, where a larger increase signals greater importance.

SliceGPT (ASHKBOOS et al., 2024) prunes models by removing components from each layer leveraging the concept of computational invariance. This principle allows for the insertion of orthogonal transformations between layers without changing the model's output. First, it constructs an orthogonal matrix by extracting eigenvectors from the product of the input matrix and its transpose for each layer. Next, it multiplies the

orthogonal matrix by a deletion matrix to eliminate selected columns, and then applies the result to the input of each block. Finally, the method multiplies the output by the transpose of the orthogonal and deletion matrices to form a new input for each layer. This approach reduces the dimensionality of the model's representation space by pruning rows and columns in the MHA and MLP blocks.

Similarly to SliceGPT, DISP-LLM (GAO et al., 2024b) applies module-wise pruning to remove rows and columns from each MHA and MLP, effectively pruning different parts of feature maps across layers. The pruning process begins by applying selection matrices and their transpose to the internal components of the Transformer to eliminate the structural dependence along the embedding dimension. In order to optimize the width of each layer in the pruned network, DISP-LLM employs ReinMax, a differentiable gradient estimator that uses a hypernetwork and latent parameters to navigate the discrete search space of possible architectures. Finally, it uses a loss-based objective function to optimize the remaining structures of the pruned model for improved performance.

MoDeGPT (LIN et al., 2025) performs the pruning process by decomposing the Multilayer Perceptron (MLP) and Multi-Head Attention (MHA) modules into subcomponents. The framework minimizes local reconstruction error by approximating each component through Nyström, CR (Column-Row), or SVD (Singular Value Decomposition) low-rank matrix factorization. This modular decomposition approach allows the model to achieve a compression rate of up to 30% with negligible increases in perplexity and significant enhancements in inference throughput.

Alternatively, PruneNet (SENGUPTA et al., 2025) uses a policy learning-based compression for LLMs. It trains a stochastic pruning policy by penalizing the Kolmogorov–Smirnov distance between the singular-value distributions of pruned and unpruned weight matrices. This penalty encourages the pruned model to preserve the original spectral properties of the weight matrices, which maintains the model's representational power. Once trained, the policy selects rows and columns to remove across target layers.

LLM Surgeon (OUDERAA et al., 2024) introduces a unified framework for structured, semi-structured, and unstructured pruning by leveraging layer-wise Kronecker-factored curvature approximations of the loss landscape, approximating larger matrices as the Kronecker product of smaller factors. The framework uses the Fisher Information Matrix to capture the second-order sensitivity of the loss function regarding the weights. First, it estimates this information via per-layer Kronecker-factored approximations. Next, it ranks individual weights, blocks, or higher-level structures according to their estimated quadratic impact on loss, enabling their removal. Finally, rather than standard gradient-based fine-tuning, it applies closed-form OBS-style weight corrections based on the Optimal Brain Surgeon algorithm (HASSIBI; STORK, 1992) between pruning shots to recover accuracy.

In contrast to the existing structured pruning approaches we mentioned above, our strategy removes attention heads and MLP neurons by projecting input data onto model weights for a small number of samples. This magnitude-based selection identifies structures with the lowest overall magnitudes, rather than relying on computationally expensive loss-based or projection-driven heuristics. Our effective design applies uniform pruning across layers, which prevents the creation of layers with varying widths, thereby avoiding additional overhead.

3.2 Semi-Structured and Unstructured Pruning

The COPAL (MALLA et al., 2024) framework identifies crucial weights by calculating the directional derivatives of the loss function across a sequence of datasets. To determine weight importance, the algorithm executes numerous differential operations by approximating the sensitivity of the model output to infinitesimal perturbations in both weights and input features for every calibration sample across the dataset sequence. This iterative process, which accumulates gradients to measure robustness across domains, significantly increases the computational cost compared to single-shot methods.

Wanda (SUN et al., 2024) identifies sparse sub-networks in LLMs by ranking each weight using the product of its absolute value and its corresponding input feature norm. This approach leverages the emergence of large-magnitude features in LLMs to determine pruning priority without requiring second-order information (DETTMERS et al., 2022). However, as Gao et al. (2024b) argue, Wanda excels only as an unstructured pruning technique, a form of sparsity that fails to yield inference speedups in the resulting model due to irregular memory access patterns.

SparseGPT (FRANTAR; ALISTARH, 2023) formulates pruning as a sparse regression and solves the objective via an efficient approximate mechanism. It uses a local layer-wise approach that minimizes the squared error between the output of the dense layer and its pruned version, employing an inverse-Hessian approximation to update remaining weights. This strategy achieves up to 60% sparsity while preserving model performance without traditional retraining. However, the mechanism yields less accurate results on small to medium-sized models compared to larger networks. Plug-and-Play (ZHANG et al., 2024) introduces a one-shot post-training pruning strategy combining Relative Importance and Activation (RIA) with Channel Permutation. RIA evaluates the significance of a weight relative to the average activation levels within its specific channel, while Channel Permutation reorders the weight matrix to satisfy structural constraints without losing high-magnitude parameters. Despite avoiding retraining, the method exhibits sensitivity to calibration data, particularly for larger models. Moreover, these methods often fail to provide practical inference speedups, as modern GPUs only accelerate inference for

semi-structured patterns (e.g., 2:4) (ZHOU et al., 2021; MISHRA et al., 2021).

Unlike semi-structured pruning methods, our technique provides practical speedups without relying on specialized hardware or following fixed sparsity patterns (e.g., 2:4 or 4:8), offering greater flexibility in reaching desired compression levels. In contrast to unstructured pruning approaches, which scatter zeroed weights throughout the model and hinder efficient computation, we achieve tangible inference speedups using a lightweight, efficient procedure.

3.3 Final Remarks

The literature reveals a trade-off between predictive ability, computational cost, and practical inference speedup. Structured approaches preserve dense model operations, but several methods rely on costly loss-based criteria or complex optimization procedures. In contrast, semi-structured and unstructured pruning achieve high sparsity levels, although practical latency gains often depend on specific sparsity patterns and specialised hardware. Our pruning technique addresses these gaps through a lightweight, data-driven structured pruning strategy, with no reliance on hardware-specific sparse operations or fixed sparsity patterns. By removing complete internal components while retaining the most relevant structures, our method targets practical inference speedup with limited degradation in predictive ability. These characteristics motivate the approach we present in the following chapter.

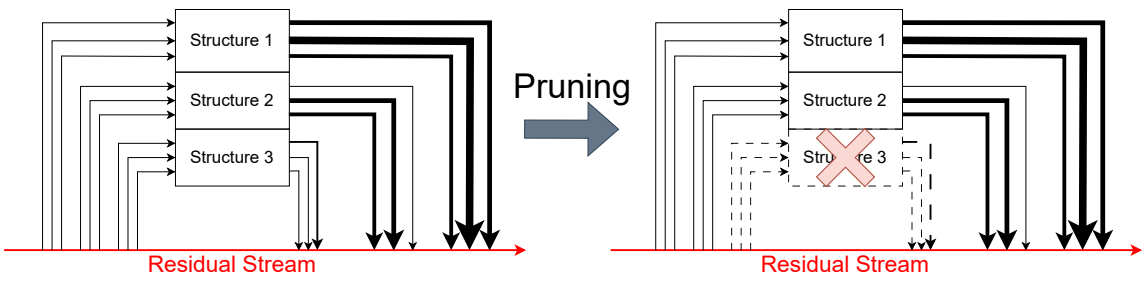
4 Proposed Method

Building upon the concept of the residual stream, our goal is to define the least important internal structures of each decoder layer of LLMs by measuring their contribution to the residual stream. As attention heads and Multilayer Perceptron (MLP) neurons occupy the majority of parameters in each layer – and consequently, the model as a whole – we consider these two structures the main targets for pruning. Hence, the name of our pruning criterion: AMP - Attention Heads and MLP Pruning.

To estimate the importance of each attention head and MLP neuron, we project the input data onto weights, checking the magnitude of their contribution to the residual stream. We determine which components to prune by summing their overall magnitude for different inputs, removing those with the lowest contributions until we reach a desired compression rate. Figure 6 shows the intuition behind the idea of removing the structure with the lowest contribution.

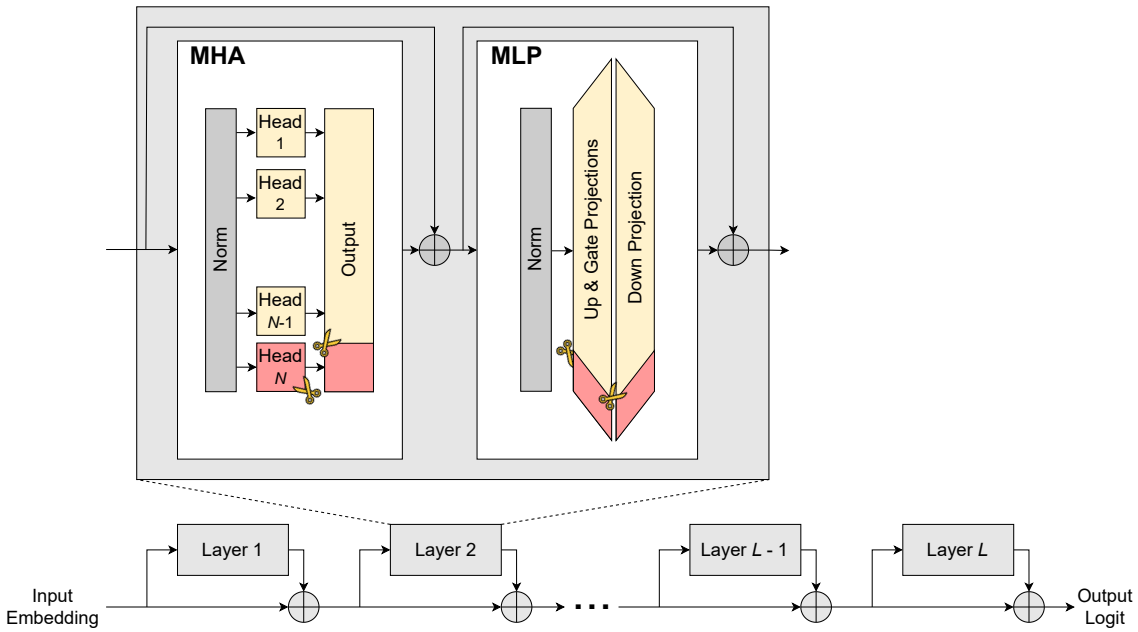
By pruning attention heads and MLP neurons, our pruning criterion offers a direct parameter count reduction, reducing computational demands. Also, by removing attention heads, we reduce the KV-Cache overhead, improving inference speedup, especially for longer input sequences. In Figure 7, we show a high-level example of how the method works. In particular, we highlight the removal of a single attention head from the MHA and neurons from the MLP projections for a specific layer. AMP automatically determines the structures to remove, surpassing existing pruning methods over multiple evaluations, as we shall see in Chapter 5.

Figure 6 – Intuition of the pruning process with AMP. By projecting input data onto weights and checking the magnitude of the contribution of each attention head and MLP neuron, our criterion determines which structures contribute the least to the residual stream, and hence, to the output of the model. In this abstraction, the width of each output arrow represents the contribution of each structure for a given input. Structure 3 contributes the least with the residual stream and becomes the first target of the pruning process.



Source: Prepared by the author.

Figure 7 – Proposed AMP. We prune attention heads and MLP neurons to achieve a target compression rate. In each of the L Transformer layers, we prune heads from the MHA module and neurons in the MLP Up-, Gate- and Down-projections (both highlighted in red). The bottom diagram shows the L layers stacked between the input embedding and the output logit. To preserve the layer-modular architecture of the Transformer, we must prune the same number of attention heads and MLP neurons from each layer.



Source: Prepared by the author.

The idea behind our method is simple; however, due to the existence of multiple notations of different models, we opt to follow the notations of one of the most popular LLMs: LLaMA-2 7B. Although the description and explanation focus on the LLaMA-2 7B model, the method is applicable to other Transformer-based LLMs.

We start with the two main components involving our AMP criterion: Multi-Head Attention (MHA) and Multilayer Perceptron (MLP).

4.1 AMP – MHA Component

The objective of the MHA component of AMP is to define an importance score for each attention head in a layer, determining which ones are less important for the overall predictive ability of the model, and hence targets for pruning.

Due to the fact that $\mathbf{W}_O \in \mathbb{R}^{(d_k \cdot N) \times d_{model}}$, the MHA output will always have size d_{model}, regardless of the number of heads N. Therefore, removing an entire attention head maintains compatibility with the dimensions of the residual stream and, consequently, with subsequent layers. To completely remove a head, we only need to prune the associated

weights from the Key, Query, and Value ($\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$) matrices. Since it reduces the dimension of the concatenated MHA output, we prune the corresponding part of $\mathbf{W}_O$ along the appropriate axis.

A naive pruning approach might extend the classical ℓ_p-norm method – common in unstructured pruning (CHENG et al., 2024) – to structured pruning by computing norms over all parameters within a single attention head. However, this strategy fails to account for the nonlinear interactions between weights and inputs. Hence, an effective metric should assess the impact of each head on the final output of MHA.

Simply measuring the output of each head $(h_1, \ldots, h_N)$ is equally insufficient, as it neglects the subsequent $\mathbf{W}_O$ projection. Additionally, evaluating only the final MHA output does not allow the scoring of each head, since the individual contributions are already combined by $\mathbf{W}_O$ at this stage (see Equation 2.8).

To address the previous limitations, the MHA component of our AMP criterion assesses the contribution of each head while incorporating the effect of the $\mathbf{W}_O$ projection. To accomplish this, we rewrite the final $\mathbf{W}_O$ linear projection and the concatenated output of the heads $\mathbf{H}$ as block matrices. Formally,

$$\mathbf{W}_O = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \quad \mathbf{W}_n \in \mathbb{R}^{d_{\text{head}} \times d_{\text{model}}}, \quad n \in \{1, 2, \ldots, N\}, \tag{4.1}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_N \end{bmatrix}, \quad \mathbf{h}_n \in \mathbb{R}^{S \times d_{\text{head}}}. \tag{4.2}$$

Then, the output of MHA becomes

$$\mathbf{O} = \mathbf{H}\mathbf{W}_o = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_N \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \tag{4.3}$$

and in a summation form

$$\mathbf{O} = \mathbf{h}_1\mathbf{W}_1 + \mathbf{h}_2\mathbf{W}_2 + \cdots + \mathbf{h}_N\mathbf{W}_N = \sum_{n=1}^{N} \mathbf{h}_n\mathbf{W}_n. \tag{4.4}$$

Overall, Equation 4.4 shows that it is possible to write the output of MHA as a sum of terms $\mathbf{h}_n\mathbf{W}_n$, where each term depends only on the output of its respective attention head. This is effectively representing the output of MHA as a sum of the contributions of each attention head.

Given the previous formalism and following previous pruning works that employ the ℓ_1-norm (HAN et al., 2015; LI et al., 2017), we define the importance I_n of each head as the ℓ_1-norm of $h_n W_n$ in terms of

$$\mathbf{I}_n = \|\mathbf{h}_n \mathbf{W}_n\|_1 . \tag{4.5}$$

Our importance criterion given by Equation 4.5 measures the magnitude of the contribution of each attention head in a layer to the final output of MHA and, hence, to the residual stream.

It is important to observe that by projecting the data onto the weights, our method differs from the common pruning strategy of simply applying the ℓ_p-norm. This distinction highlights the unique approach of measuring the importance of attention heads, setting it apart from traditional pruning techniques.

4.1.1 Importance Estimation on Grouped-Query Attention Architectures

In the Grouped-Query Attention (GQA) architecture, we partition the N logical heads into $G = N/R$ groups, with R determining the number of Queries in each group. Each group $g \in G$ keeps a single Key-Value pair $\{\mathbf{K}_g, \mathbf{V}_g\}$ but retains its own R independent query projections $\{\mathbf{Q}_{g,1}, ..., \mathbf{Q}_{g,R}\}$. For head (g, r), the Scaled Dot-Product Attention is

$$\mathbf{h}_{g,r} = \text{Attention}(\mathbf{Q}_{g,r}, \mathbf{K}_g, \mathbf{V}_g). \tag{4.6}$$

After concatenation, the output of the GQA becomes

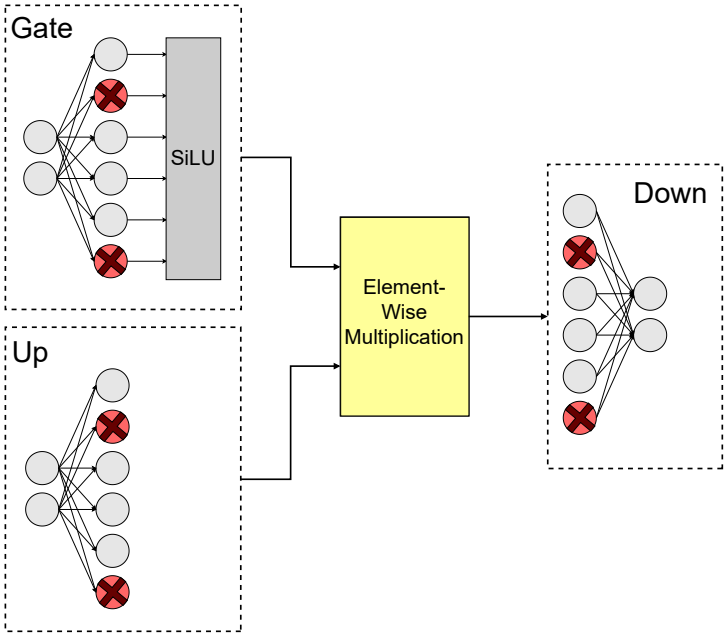
$$\mathbf{O} = \sum_{g=1}^{G} \sum_{r=1}^{R} \mathbf{h}_{g,r} \mathbf{W}_{g,r}. \tag{4.7}$$

For MHA, removing an attention head requires pruning all three of its ($\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$) matrices so that the residual stream dimension after concatenation remains d_{model}. Exactly the same dimensional constraint appears in GQA, but now one $\{\mathbf{K}_g, \mathbf{V}_g\}$ supports R different queries. Hence, for every shared $\mathbf{KV}$ we prune, we must simultaneously prune all R queries that depend on it.

Adapting the AMP criterion, we score the importance of an entire group $\mathbf{g}$ by summing the contributions of its constituent heads:

$$\mathbf{I}_g = \sum_{r=1}^{R} \|\mathbf{h}_{g,r} \mathbf{W}_{g,r}\|_1 . \tag{4.8}$$

Figure 8 – The MLP component of AMP prunes neurons in pairs from Up and Gate projections, reducing the MLP's intermediate size. Also, it prunes the neurons with the same indexes from the Down projection to maintain compatibility with the intermediate size. We highlight the idea of pruning neurons with the same indexes from all projections using an X for neurons with same indexes.



Source: Prepared by the author.

We prune the groups with the smallest importance $\mathbf{I_g}$, *i.e.* for a group g, we remove $\{\mathbf{K}_g, \mathbf{V}_g\}$ and all associated R queries, preserving the output width and the dimension of the residual stream.

4.2 AMP – MLP Component

Differently from removing attention heads, we propose pruning neurons from the Up and Gate projections in the SwiGLU MLP (see Equation 2.4), thereby reducing the MLP's intermediate size (d_i) while maintaining the hidden dimension (d_{model}). The SwiGLU mechanism employs element-wise multiplication between the Up and Gate outputs, so AMP must prune neurons in pairs from both projections. To maintain compatibility with the intermediate size d_i, we remove the neurons with the same indexes from the Down projection. We illustrate the MLP pruning process of AMP in Figure 8.

In contrast to the simple ℓ_p norm of the weights – whose score could sometimes be misleading as it ignores the input scale (FRANTAR; ALISTARH, 2023) – the MLP component of AMP accounts for both weight parameters and input magnitudes, measuring how strongly each pair of Up-Gate neurons contributes to the overall activation and, consequently, to the residual stream. Formally, for a set of input tokens x and the m-th pair of neurons, we compute the l_1-norm of element-wise product of the outputs from the

Gate and Up projections, and then average over all tokens in the sample (i.e., a sentence). This is effectively the l_1-norm of the input of the Down Projection, as we formalize in terms of

$$\mathbf{I}_m = \frac{1}{S}\sum_{s=1}^{S} \left|\text{SiLU}\left((x_s\mathbf{W}_{Gate})_m\right) \cdot (x_s\mathbf{W}_{Up})_m\right|, \tag{4.9}$$

where x_s is the MLP input of the token s and $\text{SiLU}(x_s\mathbf{W}_{Gate})_m$ represents the m-th element of the activation vector $\text{SiLU}(x_s\mathbf{W}_{Gate})$. Similarly, $(x_s\mathbf{W}_{Up})_m$ indicates the m-th element of the projection vector $x_s\mathbf{W}_{Up}$.

4.3 Overall Pruning Process with AMP

Given a Large Language Model $\mathcal{F}$ composed of a layer set L and P parameters, our goal is to remove attention heads and MLP neurons from each layer $\ell \in L$ to derive a compressed network $\mathcal{F}'$ with Q parameters, where $Q \ll P$. We define the pruning ratio c as $(1 - Q/P) \times 100$ to quantify the percentage reduction in parameters from the original model to the pruned version.

Building upon the previous problem definition, the pruning with our AMP is the following. Let $\mathbf{X}$ denote training samples, such as sentences, and $\mathbf{Y}$ the respective labels. Following previous studies (MA et al., 2023; ZHANG et al., 2024; KIM et al., 2024; SUN et al., 2024), we consider random samples from the training set $\mathbf{X}$, denoting it as $\mathbf{X}_{sub}$. In particular, these works confirm that a small calibration set is sufficient to capture key input statistics; therefore, to make a fair comparison and reduce the computational costs we follow the same practice.

Algorithm 1 summarizes the pruning process with AMP. More specifically, over steps 1 and 2 of the algorithm, $\text{AMP}(\cdot, \mathbf{X}_{sub})$ is the AMP method that extracts the mean feature representations (activations) from MLP neurons and attention heads, according to Equations 4.5 and 4.9, using the samples in $\mathbf{X}_{sub}$. Then, for each layer ℓ, steps 4 and 5 sort the importance of each attention head and MLP neurons within the layer using the activation values. To achieve a compression rate c, in steps 6 and 7, we prune c% of attention heads and c% of MLP neurons with the lowest importance uniformly across all layers, preserving the layer-modular architecture of the Transformer. After pruning across all layers, in step 9, we conduct a single fine-tuning process to recover predictive ability.

Algorithm 1 Pruning with our AMP method

Input: Large Language Model $\mathcal{F}$, Compression rate c, Training samples $\mathbf{X}$ and the respective labels $\mathbf{Y}$, Subset of samples $\mathbf{X}_{sub} \subset \mathbf{X}$

Output: $\mathcal{F}^{'}$ (Pruned version of $\mathcal{F}$)

1: $\text{AMP}_{\text{MHA}} \leftarrow \text{AMP}(\mathcal{F}, \mathbf{X}_{sub})$ ▷ Extract mean activation values for attention heads using Eq. 4.5
2: $\text{AMP}_{\text{MLP}} \leftarrow \text{AMP}(\mathcal{F}, \mathbf{X}_{sub})$ ▷ Extract mean activation values for MLP neurons using Eq. 4.9
3: **for** each layer $\ell \in L$ **do**
4: $\quad \text{AMP}_{\text{MHA}_\ell} \leftarrow \textit{sorted}(\text{AMP}_{\text{MHA}}[:, \ell])$ ▷ Sorts attention heads of layer ℓ by importance
5: $\quad \text{AMP}_{\text{MLP}_\ell} \leftarrow \textit{sorted}(\text{AMP}_{\text{MLP}}[:, \ell])$ ▷ Sorts MLP neurons of layer ℓ by importance
6: $\quad \mathcal{F}_\ell^{'} \leftarrow \mathcal{F}_\ell \setminus (\text{AMP}_{\text{MHA}_\ell}, c)$ ▷ Remove unimportant heads from ℓ based on $\text{AMP}_{\text{MHA}_\ell}$ and compression rate c
7: $\quad \mathcal{F}_\ell^{'} \leftarrow \mathcal{F}_\ell^{'} \setminus (\text{AMP}_{\text{MLP}_\ell}, c)$ ▷ Remove unimportant MLP neurons from ℓ based on $\text{AMP}_{\text{MLP}_\ell}$ and compression rate c
8: **end for**
9: Update $\mathcal{F}^{'}$ via fine-tuning on $\mathbf{X}$ and $\mathbf{Y}$

5 Experiments

We evaluate the proposed AMP criterion through a comprehensive set of experiments that analyze predictive ability, computational efficiency, and environmental impact across several LLM families. This chapter begins with the experimental settings, where we detail the selection of open-source models like LLaMA and Phi, the common-sense reasoning benchmarks, and other settings we use across all experiments. Then, we contrast our method against current state-of-the-art structured pruning techniques, showing its performance in terms of accuracy and perplexity. Further sections investigate the practical benefits of AMP, including measurable inference speedups and significant reductions in energy consumption and carbon emissions. We also evaluate AMP on LLaMA-3.3, a model with 70 billion parameters, showing the efficacy of our method on really large models. Finally, we conduct ablation studies and a coherence check to validate the effectiveness of our importance ranking and to confirm the critical roles of both the multi-head attention and multilayer perceptron components.

5.1 Experimental Settings

Large Language Models. Throughout the work, we employ six popular and open-source models: LLaMA 7B (TOUVRON et al., 2023a), LLaMA-2 7B (TOUVRON et al., 2023b), LLaMA-3.1 8B, LLaMA-3.3 70B-Instruct (GRATTAFIORI et al., 2024), Phi-1.5, and Phi-2 (LI et al., 2023). Their open-source nature makes the reproducibility of our method possible. Since pruning research extensively uses these models, this selection allows for a direct comparison between our method and existing techniques (MA et al., 2023; KIM et al., 2024; GAO et al., 2024b; ASHKBOOS et al., 2024; OUDERAA et al., 2024).

Evaluation and Datasets. Following the literature (MA et al., 2023; KIM et al., 2024), we evaluate the mean accuracy of each model at different pruning ratios for various benchmarks. For a fair comparison, we opt for common benchmarks of previous works (MA et al., 2023; KIM et al., 2024; GAO et al., 2024b; ASHKBOOS et al., 2024; OUDERAA et al., 2024). We use the EleutherAI LM Harness framework (GAO et al., 2024a) to perform zero-shot task classification on common sense reasoning datasets: WinoGrande (SAKAGUCHI et al., 2020), HellaSwag (ZELLERS et al., 2019), ARC-e / ARC-c (CLARK et al., 2018) and PIQA (BISK et al., 2020). Table 1 shows an example of a sample for each dataset. Due to the high computational demands of running these models and following common procedures of the pruning literature (MA et al., 2023; ASHKBOOS et al., 2024; SUN et al., 2024), we opt to not run statistical tests to evaluate the results. Indeed, the literature often also omits such tests because of their high computational cost, typically mitigating this limitation

by evaluating methods across a larger number of datasets. However, we highlight the deterministic nature of AMP in selecting which structures to remove, with all the possible accuracy changes coming only from the fine-tuning stage. Beyond zero-shot classification benchmarks, we perform a perplexity (PPL) analysis on WikiText-2 (MERITY et al., 2017), using the procedures of Ashkboos et al. (2024). PPL measures how well the next-token probability distribution produced by an autoregressive language model predicts an observed token sequence. Given a sequence of T tokens, $x_1, x_2, \ldots, x_T$, we define PPL as

$$\mathrm{PPL}(x_{1:T}) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{<t})\right), \tag{5.1}$$

where x_t represents the token at position t, $x_{<t}$ represents the previous tokens in the sequence, and $p_\theta(x_t \mid x_{<t})$ represents the probability attributed by the model to x_t given its preceding context. Lower PPL values indicate better probabilistic prediction quality, since the model attributes higher probabilities to the tokens that occur in the sequence. In contrast, higher values indicate greater uncertainty during next-token prediction. Unlike accuracy, PPL does not measure the correctness of discrete task predictions, but provides a complementary assessment of the language modeling capability of the model (MAGNUSSON et al., 2024).

Inference Speedup Evaluation. To evaluate inference speedup, we adopt the definition of latency from Sheng et al. (2023). Given batch size J and output length K, we define latency T as the time required to generate JK output tokens. We measure speedup as the relative difference in latency between the pruned model and its original, unpruned, counterpart. Following the procedures by Kim et al. (2024), we compute latency using 12 input tokens, 128 output tokens, and a batch size of 1 on a single GPU. We report the average results over 20 runs, excluding the initial 10 warm-up batches.

Energy and Carbon Emission Calculations. We estimate carbon emissions and energy costs associated with running the pruned models, evaluating our method from a Green AI perspective (SCHWARTZ et al., 2020; FAIZ et al., 2024; MORRISON et al., 2025). We consider the maximum energy consumption (in Watts) of an NVIDIA RTX 3090 GPU, taking into account one month of energy consumption (equivalent to 720 hours) to simulate a deployed model under continuous load. Then, we convert this energy usage into kilograms of CO_2 equivalent by applying country-specific grid emission factors. Specifically, we employ the Brazilian grid emission factor of 0.0385 kg CO_2/kWh[1] and the U.S. grid emission factor of 0.3674 kg CO_2/kWh[2]. We consider the Machine Learning Impact Calculator (LACOSTE et al., 2019) for further evaluation. We use inference speedup results to evaluate the same

[1] Ministério da Ciência, Tecnologia e Inovação. Available at: https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/noticias/2024/02/fator-de-emissao-de-co2-na-geracao-de-energia-eletrica-no-brasil-em-2023-e-o-menor-em-12-anos. Accessed: 5 June 2025.

[2] U.S. Energy Information Administration (EIA). Available at: https://www.eia.gov/tools/faqs/faq.php?id=74&t=11. Accessed: 5 June 2025.

Table 1 – Example for each common sense reasoning dataset. For all benchmarks, we input the question or context into the model with some options. The model must choose an option.

Dataset	Question / Context	Options	Correct Option
PIQA	**Goal:** How do you draw with chalk?	**Solution 1)** Melt the chalk onto pavement. **Solution 2)** Use the chalk like a pen on pavement.	2
HellaSwag	**Subject:** How to catch dragonflies. **Context:** Use a long-handled aerial net with a wide opening. Select an aerial net that is 18 inches (46cm) in diameter or larger. Look for one with a nice long handle.	**A)** Loop 1 piece of ribbon over the handle. Place the hose or hose on your net and tie the string securely. **B)** Reach up into the net with your feet. Move your body and head forward when you lift up your feet. **C)** If possible, choose a dark -colored net over a light one. Darker nets are more difficult to see, making the net more difficult to avoid. **D)** If it's not strong enough for you to handle, use a hand held net with one end shorter than the other. The net should have holes in the bottom of the net.	C
WinoGrande	Katrina had the financial means to afford a new car while Monica did not, since _____ had a high paying job	**1)** Katrina **2)** Monica	1
ARC-Challenge (ARC-C)	George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat?	**A)** dry palms **B)** wet palms **C)** palms covered with oil **D)** palms covered with lotion	A
ARC-Easy (ARC-e)	Where is most of Earth's water located?	**A)** glaciers **B)** lakes **C)** oceans **D)** rivers	C

Source: Prepared by the author.

results for the pruned model, comparing the reduction of both energy costs and carbon emissions.

Implementation Details. In order to compute the activations necessary for the method, we use 50 random samples from the cleaned version of the Alpaca dataset (TAORI et al., 2023). This choice aligns with prior studies that consider a small calibration set (typically ranging from 10 to 128 samples) to conduct the pruning process (MA et al., 2023; ZHANG et al., 2024; KIM et al., 2024; SUN et al., 2024). Following Ma et al. (2023), we employ a post-training process using low-rank adaptation, LoRA (HU et al., 2022), to recover the model performance. We set LoRA rank to 8, learning rate to 3e-4, batch size to 1, and sequence length to 512, using an AdamW optimizer. We use the Alpaca dataset and fine-tune the model for 2 epochs on a single consumer-grade GPU (NVIDIA RTX 3090 or

Table 2 – Comparison of state-of-the-art structured pruning methods across different LLMs. For each pruning ratio and model, we highlight the best results in bold. (†) For the LLaMA 7B model, the authors report an 18% pruning ratio. For a fair comparison, we reproduce the results for a closer pruning ratio (21.02% – equal to ours) using the procedures of the paper and the official code repository: https://github.com/Nota-NetsPresso/shortened-llm. In addition, since the authors do not present results for the LLaMA-2 7B model, we run their method on our own.

Pruning Ratio	Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	LLaMA 7B	69.85	76.21	72.81	44.71	79.16	68.55
20%	LLM Pruner (MA et al., 2023) (NeurIPS, 2023)	61.33	65.34	59.18	37.12	75.57	59.71
	LLM Pruner (+ fine-tuning) (MA et al., 2023) (NeurIPS, 2023)	65.11	68.11	63.43	37.88	76.44	62.19
	Shortened LLaMA (KIM et al., 2024) (ICLR, 2024) (†)	**68.82**	69.82	64.06	39.93	74.65	63.46
	DISP-LLM (GAO et al., 2024b) (NeurIPS, 2024)	64.72	68.39	64.81	37.12	76.66	62.34
	PruneNet (SENGUPTA et al., 2025) (ICLR, 2025)	62.12	65.40	64.65	36.52	75.51	60.82
	AMP (Ours)	63.93	**70.34**	**69.82**	**41.38**	**77.15**	**64.52**
0%	LLaMA-2 7B	69.14	75.99	74.58	46.15	79.11	68.99
30%	SliceGPT (ASHKBOOS et al., 2024) (ICLR, 2024)	61.33	49.62	51.77	31.23	63.55	51.50
	LLM Surgeon (OUDERAA et al., 2024) (ICLR, 2024)	61.09	60.72	63.09	36.69	73.56	59.03
	Shortened LLaMA (KIM et al., 2024) (ICLR, 2024) (†)	61.09	54.97	45.96	34.81	60.99	51.56
	DISP-LLM (GAO et al., 2024b) (NeurIPS, 2024)	**63.93**	62.87	60.1	37.03	73.72	59.53
	PruneNet (SENGUPTA et al., 2025) (ICLR, 2025)	61.09	58.30	53.20	32.94	71.11	55.33
	PruneNet (+ fine-tuning) (SENGUPTA et al., 2025) (ICLR, 2025)	62.90	63.21	53.37	33.70	72.20	57.08
	AMP (Ours)	61.25	**65.47**	**64.31**	**39.85**	**74.21**	**61.02**
0%	Phi-1.5 (1.3B)	72.77	62.58	73.11	48.04	75.63	66.43
30%	SliceGPT (ASHKBOOS et al., 2024) (ICLR, 2024)	**64.96**	42.54	53.66	31.91	65.45	51.70
	DISP-LLM (GAO et al., 2024b) (NeurIPS, 2024)	61.48	**47.97**	57.66	33.01	**71.08**	**54.24**
	AMP (Ours)	58.33	45.74	**58.38**	**35.32**	69.53	53.46
0%	Phi-2 (2.7B)	75.61	73.86	78.24	54.01	79.11	72.17
30%	SliceGPT (ASHKBOOS et al., 2024) (ICLR, 2024)	63.14	47.56	53.03	30.29	65.94	51.99
	DISP-LLM (GAO et al., 2024b) (NeurIPS, 2024)	65.19	54.43	63.59	38.48	**73.34**	59.01
	PruneNet (SENGUPTA et al., 2025) (ICLR, 2025)	**67.48**	56.80	**67.55**	**40.61**	72.80	**61.05**
	PruneNet (+ fine-tuning) (SENGUPTA et al., 2025) (ICLR, 2025)	63.93	**58.18**	61.78	37.80	71.49	58.34
	AMP (Ours)	57.22	45.89	61.03	36.69	70.95	54.36

Source: Prepared by the author.

NVIDIA RTX 4090). It is worth mentioning that this setup follows previous work (MA et al., 2023) and ensures the benefits of our method do not come from hyperparameter customizations.

5.2 Zero-shot Performance

We start our experiments by comparing the AMP criterion with top-performing structured pruning techniques. For a fair comparison, we report the results of each method according to the original paper and prune the models using the same compression rates. Table 2 summarizes the results.

Overall, our method achieves state-of-the-art accuracy across multiple tasks within the LLaMA family. Specifically, for the LLaMA 7B model, we prune 20% of its parameters and achieve the best results compared to other pruning methods. For the LLaMA-2 7B model, we reduce parameters by 30%, with only an 11.55% drop in average performance relative to the original model. These results confirm that projecting the data onto the weights helps identify the least important structures, making it possible to outperform far more computationally expensive methods, such as SliceGPT (ASHKBOOS et al., 2024)

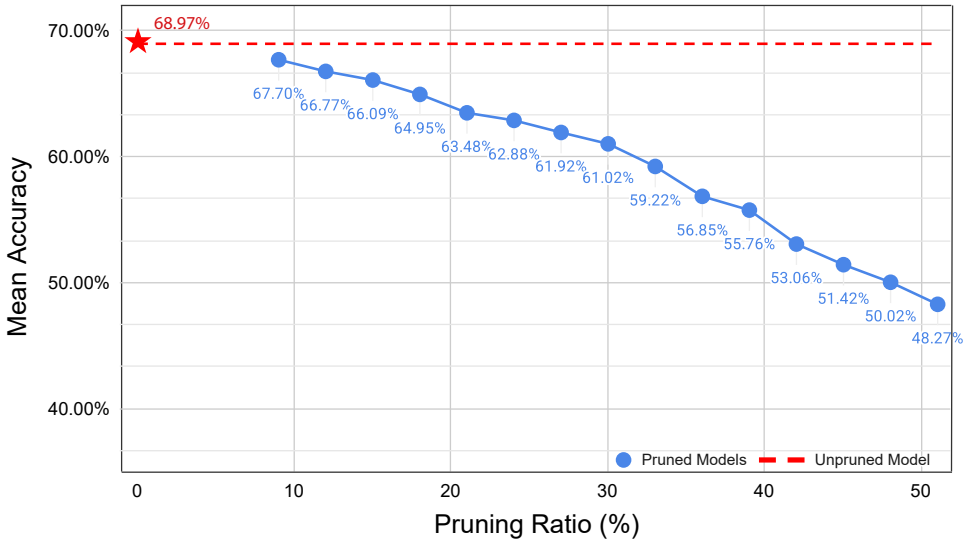
and DISP-LLM (GAO et al., 2024b), using a more efficient approach.

Although AMP delivers comparable performance for the LLaMA family, on Phi models we observe a more pronounced drop. According to Frantar & Alistarh (2023), such a result is not surprising, since smaller models generally have less capacity than larger ones and may be more sensitive to pruning, leading to greater performance declines.

The results for the WinoGrande benchmark also reveal a distinct behavior compared to other tasks, as the models we prune with AMP experience a more noticeable accuracy drop compared to models from other pruning techniques. We hypothesize that this outcome stems from the specific linguistic demands of the dataset. It turns out that WinoGrande requires the model to resolve complex binary choices based on subtle contextual cues (see Table 1). Unlike benchmarks that rely on factual retrieval, WinoGrande tasks often necessitate deeper representational nuances that our current uniform pruning ratio across layers might disrupt. Consequently, investigating why specific benchmarks such as WinoGrande resist standard compression presents a promising avenue for future research, both to refine importance metrics and to further improve the predictive ability of pruned LLMs on specific downstream tasks. We leave this question for future work, since our objective with AMP is to design a method that remains agnostic to any prior knowledge about the target dataset.

In terms of scalability, we evaluate the zero-shot performance of AMP on LLaMA-2 7B across a wide range of pruning ratios. Figure 9 summarizes the results and demonstrates a consistent and gradual decline in mean accuracy as the compression level increases from

Figure 9 – Mean accuracy across different pruning ratios for LLaMA-2 7B. We highlight the consistent and gradual decline in mean accuracy as compression increases, suggesting that AMP effectively ranks the importance of each internal component within the layers of the LLM.



Source: Prepared by the author.

0% (unpruned model) to 51.06%. Specifically, the model retains a high predictive capacity at a 9.01% pruning ratio, where the mean accuracy only drops from 68.97% to 67.70%. However, as the pruning ratio surpasses 30%, the performance deterioration becomes more evident, reaching a mean accuracy of 48.27% at the maximum compression level. This downward trend confirms that while LLMs possess significant structural redundancy, aggressive pruning eventually removes essential parameters necessary for maintaining complex reasoning capabilities across benchmarks like HellaSwag and WinoGrande. The stability of this decline suggests that AMP effectively ranks component importance, allowing for a controlled trade-off between model size and task performance throughout the entire compression spectrum.

Finally, in Figure 10, we introduce our results in terms of perplexity (PPL) on the WikiText-2 dataset. As lower PPL values indicate superior probabilistic prediction quality, the figure reveals an increasing deterioration in LLaMA-2 7B as the pruning ratio rises. This highlights the critical role of post-training in restoring model performance. In particular, the curve with fine-tuning yields significantly lower PPL values compared to the no-fine-tuning counterpart, and this gap even widens at higher pruning ratios. At a 51.06% pruning ratio, the approach without fine-tuning exhibits a PPL 6.18 times higher than the former.

As a final note, Kim et al. (2024) reported abnormally high PPL values in certain instances. In contrast, our AMP criterion exhibits consistently defined and reasonable PPL values across all compression rates. These findings reinforce its effectiveness compared to top-performance methods.

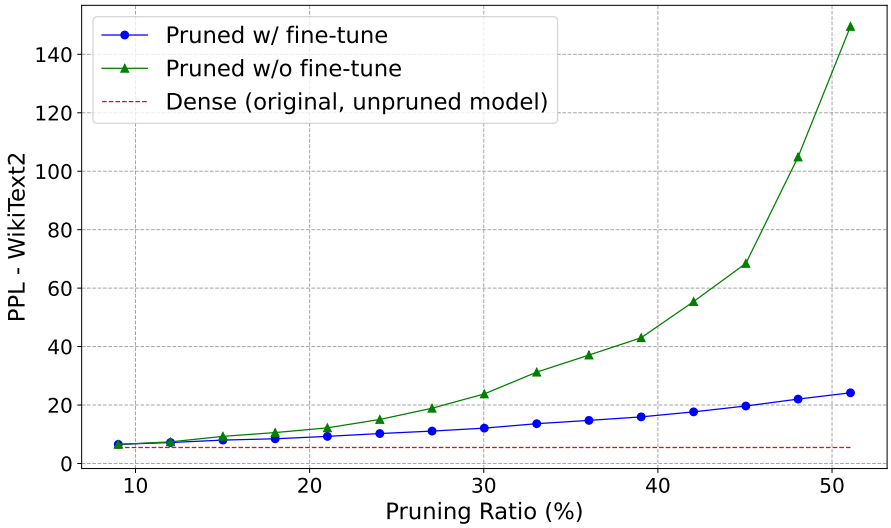
5.3 Inference Speedup

Even though the number of parameters offers theoretical insights about an LLM performance, it may fail to fully capture the model efficiency when used as a standalone metric (DEHGHANI et al., 2022). Considering this perspective, we include a latency assessment to obtain a more accurate view of the efficiency offered by our method.

We consider LLaMA 7B and LLaMA-2 7B models for inference speedup evaluation. The mean latency results are 2.90s and 2.79s for the unpruned LLaMA 7B and LLaMA-2 7B, respectively. After pruning [3], the mean latency across runs for LLaMA 7B is 2.31s, representing a speedup of 1.25×, while for LLaMA-2 7B, the mean latency is 2.34s, corresponding to a speedup of 1.19×. These results demonstrate that our method achieves practical inference speedup and is competitive with other pruning methods. In particular, we surpass methods such as Shortened LLaMA (KIM et al., 2024) and compete with semi-structured approaches like Wanda (SUN et al., 2024). The former achieves a 1.13×

[3] For both models, we consider a pruning ratio of 21.02%.

Figure 10 – Pruning impact of AMP in terms of perplexity on WikiText-2 for LLaMA-2 7B. The red dashed horizontal line indicates the performance of the original model, while the blue and green lines represent the performance of the pruned model with and without fine-tuning, respectively.



Source: Prepared by the author.

speedup by pruning entire layers and reducing model depth and the latter delivers an end-to-end inference speedup of 1.24× on LLaMA 7B (with a 2 : 4 semi-structured sparsity) by leveraging sparse tensor acceleration, thus requiring specialized hardware.

5.4 Coherence Check of Proposed AMP

Even though the results above demonstrate the competitiveness of our criterion against existing solutions, to reinforce the effectiveness of our method in selecting the least important structures for removal, we propose an evaluation to check its coherence. Instead of removing the least important structures to achieve a target pruning ratio, we remove the *most important* defined by our criterion. Intuitively, if our research statement is correct, we expect model collapse as an outcome of this reverse pruning, showing results much worse than those of the original tests. Furthermore, we also report the results for random pruning, as previous works suggest its effectiveness (WANG et al., 2022; LI et al., 2022). In this experiment, we use the LLaMA-2 7B model at three compression rates across different benchmarks.

Table 3 shows the results and supports that the proposed metric effectively identifies and ranks less critical structures for pruning. It turns out that when the method is reversed – meaning the most important components are pruned – the results degrade significantly, with an accuracy drop of more than 25 percentage points (pp) across all pruning ratios compared to the original ranking. Interestingly, random pruning, although less effective

Table 3 – Coherence check of the AMP method. We highlight in bold the best average accuracy across tasks for each pruning ratio. Based on the results, we confirm that our method effectively identifies and removes structures of lower importance.

Pruning Ratio	Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	LLaMA-2 7B	69.06	76.02	74.54	46.16	79.05	68.97
9.01%	**AMP**	65.98	75.02	72.81	47.10	77.58	**67.70**
	Random	59.27	69.63	65.74	41.13	76.17	62.39
	Reversed	49.25	26.55	26.85	27.47	49.89	36.00
21.02%	**AMP**	61.56	69.22	68.18	42.06	76.39	**63.48**
	Random	58.72	62.83	54.21	35.92	73.18	56.97
	Reversed	47.91	26.32	26.89	27.05	50.49	35.73
30.03%	**AMP**	61.25	65.47	64.31	39.85	74.21	**61.02**
	Random	55.25	53.94	50.55	30.89	68.82	51.89
	Reversed	48.22	26.28	25.51	27.56	49.67	35.45

Source: Prepared by the author.

than our original method, still achieves better performance than the reversed-importance approach.

In summary, these results reinforce that AMP successfully differentiates between essential and non-essential structures, validating the strategy of selecting lower-importance elements for pruning. Most importantly, this experiment corroborates our research statement by showing that we effectively estimate the importance of internal components by projecting input data onto weights.

We believe this experiment is essential for validating any pruning criteria, as it is simple and mitigates the influence of hyperparameter variations and other confounding factors, and we hope to see future efforts on pruning taking it into account.

5.5 The Role of the MHA and MLP Components in AMP

To evaluate the individual contributions of the attention head and MLP components within AMP, in this experiment, we conduct an ablation study by separately pruning attention heads and MLP neurons, and compare the results against applying the full AMP criterion that eliminates both components. We follow the same setup as in Subsection 5.2.

Table 4 exhibits the results for a 30% compression rate on LLaMA-2 7B. According to this table, we observe that pruning only attention heads severely impairs predictive performance, reducing the average accuracy by 23.39 pp compared to pruning both attention heads and MLPs. It turns out that, for all the models we analyze, MHA accounts for approximately 33% of the total parameters in a layer. Thus, achieving a compression rate of 30% by pruning only attention heads would require removing nearly all of them — for LLaMA-2 7B, 30 out of the 32 heads in each layer — leading to large performance drops. On the other hand, pruning MLP neurons provides a more effective reduction

Table 4 – Influence of removing isolated components and all at once, using LLaMA-2 7B with a 30% compression rate. We highlight in bold the best average accuracy across tasks. The results emphasize that AMP, when pruning both heads and MLP neurons (first row), is substantially better than the removal of only one of them.

Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
AMP (Removing MHA + MLP)	61.25	65.47	64.31	39.85	74.21	**61.02**
AMP (Removing only MLP)	57.14	60.64	59.81	34.64	72.42	56.93
AMP (Removing only MHA)	49.49	28.13	30.60	24.23	55.71	37.63

Source: Prepared by the author.

in model size compared to pruning attention heads alone, as these neurons constitute approximately 67% of the parameters within a layer. However, our experiment shows that the removal of only MLP neurons also leads to a decrease in performance, with an average accuracy drop of 4.09 pp compared to pruning both components together.

Overall, by removing all components together, AMP ensures that no component is removed prematurely, allowing higher compression rates while maintaining predictive ability.

5.6 Effectiveness of AMP on GQA models

As we explain in Subsection 2.1.4.1, LLaMA-3.1 8B (GRATTAFIORI et al., 2024) reduces the KV cache overhead by replacing MHA with Grouped-Query Attention (GQA). Instead of having 32 KV heads, such as LLaMA-2 7B, the official architecture divides the number of heads, keeping only 8 KV heads, each one attending to 4 different Query heads. This reduces both the KV cache size and read bandwidth fourfold. We adapt the AMP importance criterion (Eq. 4.5) to take GQA into account and refer interested readers to the complete formalism in Subsection 4.1.1.

Table 5 shows that pruning reduces the accuracy of LLaMA-3.1 8B across benchmarks to approximately 35%, close to the random guess mean accuracy[4]. Lin et al. (2025) demonstrate that **KV** matrices are more sensitive to pruning. We also hypothesize that the low number of **KV** matrices in GQA – just eight in the case of LLaMA-3.1 8B – is likely a determining factor that makes these structures even more essential for model performance.

Building on these insights, we modify AMP to prune only the MLP neurons while retaining all attention heads. We compare the results with MoDeGPT (LIN et al., 2025), the current state-of-the-art pruning technique for LLaMA-3.1. Table 6 introduces the results. Overall, removing only MLP neurons from the model reduces the performance

[4] Random-guess accuracy of each benchmark: 50% on WinoGrande and PIQA; 25% on HellaSwag, ARC-e, and ARC-c. By averaging these values, we compute the random guess mean accuracy, which is 35%.

Table 5 – Comparison of the original LLaMA-3.1 8B vs. its pruned counterpart. The results emphasize that AMP, combining the removal of attention heads and MLP neurons, drops the accuracy closer to random guess (mean accuracy of 35% over the benchmarks) for the LLaMA-3.1 8B model.

Pruning Ratio	Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	LLaMA-3.1 8B	74.19	79.05	81.23	53.16	81.12	73.75
30%	AMP	53.35	27.19	28.66	22.87	53.48	37.11

Source: Prepared by the author.

Table 6 – Comparison between the adapted version of AMP vs. state-of-the-art pruning method for LLaMA-3.1 8B. We highlight that we attempted to contact the authors of MoDeGPT (LIN et al., 2025) to reproduce their work; however, due to company copyright restrictions, they were unable to provide their implementation.

Pruning Ratio	Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	LLaMA-3.1 8B	74.19	79.05	81.23	53.16	81.12	73.75
25%	AMP (Removing only MLP)	61.96	65.35	60.65	39.93	73.94	60.37
	MoDeGPT (LIN et al., 2025) (ICLR, 2025)	69.61	66.49	67.05	41.13	75.52	**63.96**
30%	AMP (Removing only MLP)	61.40	60.49	60.35	37.20	71.98	**58.28**
	AMP (Removing GQA + MLP)	53.35	27.19	28.66	22.87	53.48	37.11

Source: Prepared by the author.

drop compared to the original AMP method, which prunes both MHA and MLP in the same proportion. When we compare the results with MoDeGPT, the adapted version of AMP is still competitive, with a difference of only 3.59 pp between the two techniques for a compression rate of 25%. It is worth mentioning that, compared to our criterion, MoDeGPT is a much more expensive method that relies on multiple low-rank matrix approximation techniques.

5.7 Impact on the Number of Fine-tuning Epochs

Pruning an LLM removes parameters and reduces its representational capacity, often reducing predictive ability. By conducting the fine-tuning process, we seek to recover some of the lost capabilities (LE; HUA, 2021), but the optimal number of fine-tuning epochs remains unclear. Although a short fine-tuning schedule may under-adapt the pruned model and degrade task performance, a long schedule increases computational cost and often yields diminishing returns on a small calibration set. To study how training duration affects recovery, we fine-tune the LLaMA-2 7B model at compression rates of 21.02% and 30.03% over 1 to 4 epochs. We employ the same hyperparameters and benchmarks as those in Subsection 5.2.

For a compression rate of 21.02%, the average results across tasks are 64.12%, 63.44%, 63.41% and 62.56% for 1, 2, 3, and 4 epochs of fine-tuning, respectively. For a compression ratio of 30.03%, the results are 60.72%, 61.02%, 60.10% and 59.18%. These

results indicate that more than one epoch does not yield significant improvement. We therefore adopt two epochs in all experiments to ensure a fair comparison with prior work such as LLM-Pruner (MA et al., 2023).

5.8 Pruning a 70B-Parameter Model

To further evaluate the efficacy of AMP across model scales, we also prune the LLaMA-3.3 70B model [5]. In fact, while an RTX 4090 suffices for smaller architectures, the 70B parameter count demands higher memory capacity. Consequently, we rent an NVIDIA H100 GPU with 96GB of VRAM. This shows that even though pruning methods make models more accessible for inference, the pruning process involves significant costs depending on the scale of the model. Specifically, loading the unpruned model and running the fine-tuning process represent the most compute-intensive steps in the pipeline. Due to these high costs, many state-of-the-art techniques opt to not prune models with this scale.

Nevertheless, even with a rented GPU, we still face computational constraints. To address the memory limitation, we use the quantized 4-bit version of LLaMA-3.3 70B, reducing the precision of each parameter from 16 bits to 4 bits, which lowers the GPU memory consumption by fourfold.

LLaMA-3.3 70B adopts a more aggressive GQA configuration than LLaMA-3.1 8B. While LLaMA-3.1 8B shares four Query heads for each KV head, LLaMA-3.3 70B shares eight Query heads for each KV head. From Subsection 5.6, we show the removal of heads in GQA architectures makes the predictive ability of the model close to random guess. Since LLaMA-3.3 uses an even larger number of shared heads, we expect head removal to produce an even less favorable trade-off between compression and accuracy. For this reason, we prune only the MLP neurons and keep the attention structure unchanged.

In addition, as we discuss in Subsection 5.7, varying the number of fine-tuning epochs on the Alpaca dataset does not appear to substantially influence the overall performance. Since running LLaMA-3.3 70B remains very costly, we fine-tune the pruned model for only one epoch.

We opt to remove 50% of the MLP units, reducing the intermediate size from 28672 to 14336. This corresponds to an overall compression rate of 39.95%. Table 7 reports the results on the commonsense reasoning benchmarks and compares LLaMA-3.3 70B with other unpruned models.

Overall, the original LLaMA-3.3 70B reaches 78.25% average accuracy, clearly outperforming all smaller baselines in the table. After pruning, but before fine-tuning, the compressed model attains 67.75% average accuracy, a drop of 10.50 percentage points.

[5] Model available at: https://huggingface.co/unsloth/Llama-3.3-70B-Instruct-bnb-4bit. Accessed: 21 March 2026.

Table 7 – Comparison between LLaMA-3.3 70B against smaller unpruned models. The results highlight the efficacy of AMP in preserving the predictive ability of the pruned model and show that, for this specific set of benchmarks, pruning a very large model does not yield the most favorable trade-off between accuracy and computational cost, as smaller unpruned models with only a fraction of the parameters achieve similar or even superior performance.

Pruning Ratio	Model	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	Phi-1.5 (1.3B)	72.77	62.58	73.11	48.04	75.63	66.43
	Phi-2 (2.7B)	75.61	73.86	78.24	54.01	79.11	72.17
	LLaMA 7B	69.85	76.21	72.81	44.71	79.16	68.55
	LLaMA-2 7B	69.06	76.02	74.54	46.16	79.05	68.97
	LLaMA-3.1 8B	74.19	79.05	81.23	53.16	81.12	73.75
	LLaMA-3.3 70B	**79.16**	**84.19**	**83.04**	**61.35**	**83.51**	**78.25**
39.95%	LLaMA-3.3 70B (without fine-tuning)	67.96	71.15	72.60	50.60	76.44	67.75
	LLaMA-3.3 70B (with fine-tuning)	70.72	75.84	73.78	51.71	77.91	69.99

Source: Prepared by the author.

A single epoch of fine-tuning recovers part of this loss and raises the mean accuracy to 69.99%, reducing the gap to the original model to 8.26 percentage points. These results show that AMP remains effective even at the 70B scale, although aggressive compression still affects predictive ability.

Furthermore, we also highlight that, for the benchmarks we consider, pruning LLaMA-3.3 70B does not provide the most favorable practical trade-off. Although pruning reduces the parameter count substantially, the resulting model remains extremely large, with more than 42 billion parameters. At the same time, its average accuracy after fine-tuning reaches 69.99%, remaining closer to or below much smaller dense baselines. In particular, the pruned and fine-tuned LLaMA-3.3 70B surpasses LLaMA 7B (68.55%), LLaMA-2 7B (68.97%), and Phi-1.5 (66.43%), but still underperforms Phi-2 (72.17%) and LLaMA-3.1 8B (73.75%), much smaller models. Therefore, for these commonsense reasoning tasks, using a very large model may not be justified, even after pruning, since smaller unpruned models offer a more attractive balance between predictive ability and computational cost.

Overall, the previous results suggest that the benefits of pruning depend strongly on the downstream task: in scenarios where smaller dense models already perform competitively, they may constitute a more appropriate choice than a heavily pruned yet still extremely large model.

5.9 Green AI and Financial Costs

The concept of Green AI has gained attention among researchers, emphasizing the necessity of lowering the computational demands associated with developing and deploying AI models, aiming to minimize environmental impact (SCHWARTZ et al., 2020; FAIZ et al., 2024; MORRISON et al., 2025). In particular, LLMs are well-known for their

high computational costs, both in training and deployment, requiring increasingly large computational resources due to the scaling laws governing their development (HOFFMANN et al., 2022).

Our technique aligns with this concept, significantly lowering the computational costs of LLMs. AMP achieves up to 1.25× inference speedup, representing a 20% direct decrease in operational costs. Furthermore, our method provides a reduction of 1.94 kilograms of equivalent CO_2, the same as 7.9 kilometers of a common passenger car. It is important to mention that most of the Brazilian energy comes from renewable sources [6]. In contrast, if we consider the carbon efficiency of the USA, a country with less than 10% of energy production coming from renewable sources [7] and where most of the AI datacenters are, the reduction in emissions is 18.51 kilograms of equivalent CO_2, the same as 75 kilometers of a passenger car.

Furthermore, the complete pruning and post-training processes of AMP take approximately four hours to complete using a GPU RTX 3090, with the pruning process itself consuming only a few minutes. These computational costs are typically negligible when compared to the expenses of running these models in production over extended periods (MORRISON et al., 2025; MASLEJ et al., 2024; MASLEJ et al., 2025).

We highlight that AI providers, such as OpenAI, leverage the use of not one, but tens of thousands of GPUs. In this scenario, we emphasize the importance of techniques that lower the computational costs, such as pruning, to minimize environmental impacts.

[6] Ministério de Minas e Energia. Available at: https://www.gov.br/mme/pt-br/assuntos/noticias/fontes-renovaveis-responderam-por-93-1-da-geracao-de-energia-eletrica-em-2023. Accessed: 5 June 2025.

[7] U.S. Energy Information Administration (EIA). Available at: https://www.eia.gov/energyexplained/us-energy-facts/data-and-statistics.php. Accessed: 5 June 2025.

6 Conclusions

The high computational costs and slow latency of LLMs hinder their applicability in resource-constrained environments. In this work, we propose an effective technique to reduce computational demands and enhance the performance of pruned LLMs. Our strategy, called AMP, identifies and eliminates the least critical components (attention heads and MLP neurons) from LLMs. Through extensive experiments, we show that, by projecting input data onto weights, we successfully quantify the importance of the internal components of LLMs, producing models that are lighter, faster, and maintain their predictive capacity. In particular, this addresses our first research question of how to systematically identify and prune the least relevant structures in LLMs.

When we compare our method with state-of-the-art structured pruning approaches, our AMP criterion surpasses existing techniques by a large margin, achieving mean accuracy improvements of up to 4.81 pp for LLaMA 7B and up to 9.52 pp for LLaMA-2 7B. Furthermore, the perplexity evaluation indicates that the pruned models predict the next tokens with high probability levels, demonstrating the consistency of AMP across many compression rates. In particular, AMP achieves up to 1.25× inference speedup without requiring specialized hardware, further enhancing efficiency and sustainability. In addition to these benefits, our method aligns with Green AI principles, decreasing the carbon footprint and energy consumption of LLMs, emphasizing the role of AMP in reducing the environmental impact of deploying these models. Overall, these results characterize the trade-offs of pruning in terms of model accuracy, inference latency, energy use, environmental footprint, and associated financial costs, thereby addressing our second research question.

Our results also answer the third research question of how removing different internal components of LLMs influences predictive ability. In LLaMA-2, ablation studies show that removing both attention heads and MLP neurons yields better performance than removing each component in isolation, indicating that these structures contribute differently to the final predictive ability of the model. We also present evidence of the importance of the post-training stage in restoring model performance after pruning, corroborating previous work (XIA et al., 2024). By introducing a coherence check experiment, we confirm that AMP correctly ranks the importance of each component. In particular, when our method removes components ranked as more critical, namely those it should preserve, the predictive ability of the model drastically decreases. We consider this verification essential for validating any pruning criterion, and we hope that it becomes a default validation for future pruning research.

Finally, we also address the fourth and final research question of how architectural differences across LLMs create challenges for designing model-agnostic pruning techniques. In contrast to LLaMA-2, LLaMA-3.1 and LLaMA-3.3 adopt Grouped-Query Attention instead of Multi-Head Attention; therefore, we prune only MLP neurons in these models while preserving all attention heads, achieving substantial compression with minimal accuracy degradation. This divergence illustrates that pruning strategies tailored to the unique mechanisms of each architecture deliver superior performance–efficiency trade-offs compared with one-size-fits-all approaches, and that effective pruning must be architecture-driven and model-specific.

For ease of access, we reiterate the reproducibility resources we show in Chapter 1: the source code is available at https://github.com/c2d-usp/Efficient-LLMs-with-AMP, and the model checkpoints at https://huggingface.co/collections/c2d-usp/efficient-large-language-models.

6.1 Limitations and Future Work

Despite these contributions, the present work has methodological and experimental limitations. First, our evaluation focuses mainly on common-sense reasoning benchmarks, complemented by perplexity on WikiText-2. Therefore, the conclusions remain dependent on the tasks and evaluation settings considered in this dissertation. We also do not perform statistical tests on predictive-performance results due to the computational cost of repeated LLM evaluations. Future studies should extend the analysis to other tasks and domains, include repeated experiments when computationally feasible, and investigate generation settings such as different temperatures and context lengths. Evaluating factuality and hallucination also remains important to determine whether pruning affects model behaviour beyond accuracy and perplexity.

The current formulation of AMP also introduces methodological choices that require further investigation. AMP applies the same pruning rate across Transformer layers, although different portions of the network may present different sensitivities to pruning. An adaptive strategy could therefore assign compression rates according to layer importance. Similarly, although the use of the ℓ_1-norm follows established pruning literature, we do not experimentally compare it against ℓ_2 or other aggregation strategies. Such comparisons represent an important ablation for future work. Architectural differences also affect the structures that AMP removes. In particular, the sensitivity of Grouped-Query Attention to head removal leads us to preserve the attention structure of LLaMA-3.1 and LLaMA-3.3, indicating that effective pruning remains dependent on characteristics of each model architecture. Extending AMP to multimodal architectures represents another relevant direction.

Finally, the computational requirements of the compression pipeline remain an important limitation. AMP relies on post-pruning fine-tuning to recover predictive ability, introducing additional computational, financial, and energy costs. This issue becomes more relevant at larger scales, as we show in the LLaMA-3.3 70B experiment, in which the original model still requires high-memory hardware before compression. Moreover, our results show that pruning a very large model does not necessarily provide the most favorable performance–efficiency trade-off, since smaller dense models may offer superior results for specific downstream tasks. Future work should therefore investigate lower-cost recovery strategies, different fine-tuning configurations and task-specific adaptations, while considering the target task when deciding between pruning a large model and adopting a smaller dense architecture.

Bibliography

ABDIN, M. et al. Phi-3 technical report: A highly capable language model locally on your phone. *ArXiv*, 2024. Available on: https://arxiv.org/abs/2404.14219. Cited in page 23.

AINSLIE, J. et al. GQA: training generalized multi-query transformer models from multi-head checkpoints. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. Cited 2 times in pages 23 and 24.

ASHKBOOS, S. et al. SliceGPT: Compress large language models by deleting rows and columns. *International Conference on Learning Representations (ICLR)*, 2024. Cited 5 times in pages 13, 30, 41, 42, and 44.

BA, L. J.; KIROS, J. R.; HINTON, G. E. Layer normalization. *ArXiv*, 2016. Available on: https://arxiv.org/abs/1607.06450. Cited in page 24.

BISK, Y. et al. Piqa: Reasoning about physical commonsense in natural language. *AAAI Conference on Artificial Intelligence*, 2020. Cited in page 41.

BROWN, T. B. et al. Language models are few-shot learners. *Neural Information Processing Systems (NeurIPS)*, 2020. Cited 4 times in pages 12, 25, 26, and 27.

CHANG, X. et al. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. *AAAI Conference on Artificial Intelligence*, 2021. Cited in page 29.

CHEN, S.; ZHAO, Q. Shallowing deep networks: Layer-wise pruning based on feature representations. *IEEE Trans. Pattern Anal. Mach. Intell. (IEEE TPAMI)*, 2019. Cited in page 13.

CHENG, H.; ZHANG, M.; SHI, J. Q. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans. Pattern Anal. Mach. Intell. (IEEE TPAMI)*, 2024. Cited 2 times in pages 13 and 36.

CHO, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. Cited in page 19.

CHOWDHERY, A. et al. Palm: Scaling language modeling with pathways. *ArXiv*, 2022. Available on: https://arxiv.org/abs/2204.02311. Cited in page 23.

CLARK, P. et al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, 2018. Available on: https://arxiv.org/abs/1803.05457. Cited in page 41.

DEEPSEEK-AI. Deepseek-v3 technical report. *ArXiv*, 2024. Available on: https://arxiv.org/abs/2412.19437. Cited 2 times in pages 13 and 27.

DEHGHANI, M. et al. The efficiency misnomer. *International Conference on Learning Representations (ICLR)*, 2022. Cited in page 46.

DETTMERS, T. et al. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Neural Information Processing Systems (NeurIPS)*, 2022. Cited in page 32.

DETTMERS, T. et al. Qlora: Efficient finetuning of quantized llms. *Neural Information Processing Systems (NeurIPS)*, 2023. Cited in page 26.

DING, N. et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 2023. Cited in page 26.

ELHAGE, N. et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. Available on: https://transformer-circuits.pub/2021/framework/index.html. Cited 2 times in pages 24 and 25.

FAIZ, A. et al. Llmcarbon: Modeling the end-to-end carbon footprint of large language models. *International Conference on Learning Representations (ICLR)*, 2024. Cited 2 times in pages 42 and 52.

FRANTAR, E.; ALISTARH, D. SparseGPT: Massive language models can be accurately pruned in one-shot. *International Conference on Machine Learning (ICML)*, 2023. Cited 4 times in pages 13, 32, 38, and 45.

FRANTAR, E. et al. OPTQ: Accurate quantization for generative pre-trained transformers. *International Conference on Learning Representations (ICLR)*, 2023. Cited in page 13.

GANJDANESH, A.; GAO, S.; HUANG, H. Jointly training and pruning cnns via learnable agent guidance and alignment. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. Cited in page 29.

GAO, L. et al. *The Language Model Evaluation Harness*. Zenodo, 2024. Available on: https://zenodo.org/records/12608602. Cited 2 times in pages 28 and 41.

GAO, S. et al. DISP-LLM: Dimension-independent structural pruning for large language models. *Neural Information Processing Systems (NeurIPS)*, 2024. Cited 6 times in pages 13, 31, 32, 41, 44, and 45.

GAO, T.; FISCH, A.; CHEN, D. Making pre-trained language models better few-shot learners. *Association for Computational Linguistics (ACL)*, 2021. Cited in page 27.

GRATTAFIORI, A. et al. The llama 3 herd of models. *ArXiv*, 2024. Available on: https://arxiv.org/abs/2407.21783. Cited 6 times in pages 13, 23, 25, 27, 41, and 49.

GROMOV, A. et al. The unreasonable ineffectiveness of the deeper layers. *International Conference on Learning Representations (ICLR)*, 2025. Cited in page 14.

HAN, S. et al. Learning both weights and connections for efficient neural networks. *Neural Information Processing Systems (NeurIPS)*, 2015. Cited in page 37.

HASSIBI, B.; STORK, D. G. Second order derivatives for network pruning: optimal brain surgeon. *Neural Information Processing Systems (NeurIPS)*, 1992. Cited in page 31.

HE, K. et al. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited in page 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, 1997. Cited in page 19.

HOFFMANN, J. et al. Training compute-optimal large language models. *Neural Information Processing Systems (NeurIPS)*, 2022. Cited 3 times in pages 25, 26, and 53.

HU, E. J. et al. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022. Cited 4 times in pages 25, 26, 29, and 43.

HU, P. et al. OPQ: compressing deep neural networks with one-shot pruning-quantization. *AAAI Conference on Artificial Intelligence*, 2021. Cited in page 13.

JIANG, A. Q. et al. Mistral 7b. *ArXiv*, 2023. Available on: https://arxiv.org/abs/2310.06825. Cited in page 23.

JIN, R. et al. A comprehensive evaluation of quantization strategies for large language models. *Association for Computational Linguistics (ACL)*, 2024. Cited in page 13.

JÓZEFOWICZ, R.; ZAREMBA, W.; SUTSKEVER, I. An empirical exploration of recurrent network architectures. *International Conference on Machine Learning (ICML)*, 2015. Cited in page 19.

KIM, B. et al. Shortened llama: A simple depth pruning for large language models. *International Conference on Learning Representations (ICLR) - Workshop*, 2024. Cited 8 times in pages 13, 30, 39, 41, 42, 43, 44, and 46.

KOTHA, S.; SPRINGER, J. M.; RAGHUNATHAN, A. Understanding catastrophic forgetting in language models via implicit inference. *International Conference on Learning Representations (ICLR)*, 2024. Cited in page 29.

KUDO, T. Subword regularization: Improving neural network translation models with multiple subword candidates. *Association for Computational Linguistics (ACL)*, 2018. Cited in page 19.

LACOSTE, A. et al. Quantifying the carbon emissions of machine learning. *Neural Information Processing Systems (NeurIPS)*, 2019. Cited in page 42.

LE, D. H.; HUA, B. Network pruning that matters: A case study on retraining variants. *International Conference on Learning Representations (ICLR)*, 2021. Cited in page 50.

LI, H. et al. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*, 2017. Cited in page 37.

LI, Y. et al. Revisiting random channel pruning for neural network compression. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited in page 47.

LI, Y. et al. Textbooks are all you need II: phi-1.5 technical report. *ArXiv*, 2023. Available on: https://arxiv.org/abs/2309.05463. Cited in page 41.

LIN, C. et al. Modegpt: Modular decomposition for large language model compression. *International Conference on Learning Representations (ICLR)*, 2025. Cited 3 times in pages 31, 49, and 50.

LINDSEY, J. et al. On the biology of a large language model. *Transformer Circuits Thread*, 2025. Available on: https://transformer-circuits.pub/2025/attribution-graphs/biology.html. Cited in page 24.

MA, X.; FANG, G.; WANG, X. Llm-pruner: On the structural pruning of large language models. *Neural Information Processing Systems (NeurIPS)*, 2023. Cited 8 times in pages 13, 29, 30, 39, 41, 43, 44, and 51.

MAGNUSSON, I. et al. Paloma: A benchmark for evaluating language model fit. *Neural Information Processing Systems (NeurIPS)*, 2024. Cited in page 42.

MALLA, S.; CHOI, J. H.; CHOI, C. COPAL: Continual pruning in large language generative models. *International Conference on Machine Learning (ICML)*, 2024. Cited in page 32.

MASLEJ, N. et al. Artificial intelligence index report 2024. *ArXiv*, 2024. Available on: https://arxiv.org/abs/2405.19522. Cited in page 53.

MASLEJ, N. et al. Artificial intelligence index report 2025. *ArXiv*, 2025. Available on: https://arxiv.org/abs/2504.07139. Cited 2 times in pages 14 and 53.

MERITY, S. et al. Pointer sentinel mixture models. *International Conference on Learning Representations (ICLR)*, 2017. Cited in page 42.

MIKOLOV, T. et al. Learning longer memory in recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2015. Cited in page 19.

MISHRA, A. K. et al. Accelerating sparse deep neural networks. *ArXiv*, 2021. Available on: https://arxiv.org/abs/2104.08378. Cited in page 33.

MITTAL, D. et al. Recovering from random pruning: On the plasticity of deep convolutional neural networks. *Winter Conference on Applications of Computer Vision (WACV)*, 2018. Cited in page 29.

MORRISON, J. et al. Holistically evaluating the environmental impact of creating language models. *International Conference on Learning Representations (ICLR)*, 2025. Cited 4 times in pages 13, 42, 52, and 53.

MUGNAINI, L. G. et al. Layer pruning with consensus: A triple-win solution. *IEEE Access*, 2025. Cited in page 18.

MUGNAINI, L. G. et al. Efficient llms with amp: Attention heads and mlp pruning. *International Joint Conference on Neural Networks (IJCNN)*, 2025. Cited in page 18.

MURALIDHARAN, S. et al. Compact language models via pruning and knowledge distillation. *Neural Information Processing Systems (NeurIPS)*, 2024. Cited in page 29.

NASCIMENTO, G. H. do et al. Pruning everything, everywhere, all at once. *International Joint Conference on Neural Networks (IJCNN)*, 2025. Cited in page 29.

OPENAI. GPT-4 technical report. *ArXiv*, 2023. Available on: https://arxiv.org/abs/2303.08774. Cited in page 12.

OUDERAA, T. F. A. van der et al. The LLM surgeon. *International Conference on Learning Representations (ICLR)*, 2024. Cited 4 times in pages 13, 31, 41, and 44.

OUYANG, L. et al. Training language models to follow instructions with human feedback. *Neural Information Processing Systems (NeurIPS)*, 2022. Cited in page 25.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2013. Cited in page 19.

PONS, I. et al. Effective layer pruning through similarity metric perspective. *International Conference on Pattern Recognition (ICPR)*, 2024. Cited in page 29.

POPE, R. et al. Efficiently scaling transformer inference. *Conference on Machine Learning and Systems (MLSys)*, 2023. Cited in page 23.

SAEEDI, A. et al. Knowledge distillation via constrained variational inference. *AAAI Conference on Artificial Intelligence*, 2022. Cited in page 13.

SAKAGUCHI, K. et al. Winogrande: An adversarial winograd schema challenge at scale. *AAAI Conference on Artificial Intelligence*, 2020. Cited in page 41.

SALINAS, D. et al. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2020. Cited in page 19.

SCHWARTZ, R. et al. Green AI. *Association for Computing Machinery (ACM)*, 2020. Cited 3 times in pages 13, 42, and 52.

SENGUPTA, A.; CHAUDHARY, S.; CHAKRABORTY, T. You only prune once: Designing calibration-free model compression with policy learning. *International Conference on Learning Representations (ICLR)*, 2025. Cited 2 times in pages 31 and 44.

SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural machine translation of rare words with subword units. *Association for Computational Linguistics (ACL)*, 2016. Cited in page 19.

SHAZEER, N. Fast transformer decoding: One write-head is all you need. *ArXiv*, 2019. Available on: https://arxiv.org/abs/1911.02150. Cited in page 23.

SHEN, M. et al. Structural pruning via latency-saliency knapsack. *Neural Information Processing Systems (NeurIPS)*, 2022. Cited in page 29.

SHENG, Y. et al. Flexgen: High-throughput generative inference of large language models with a single GPU. *International Conference on Machine Learning (ICML)*, 2023. Cited in page 42.

SKEAN, O. et al. Layer by layer: Uncovering hidden representations in language models. *International Conference on Machine Learning (ICML)*, 2025. Cited in page 24.

SRIVASTAVA, A. et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. Cited in page 28.

SU, J. et al. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. Cited in page 22.

SUN, M. et al. A simple and effective pruning approach for large language models. *International Conference on Learning Representations (ICLR)*, 2024. Cited 7 times in pages 13, 29, 32, 39, 41, 43, and 46.

TAORI, R. et al. Stanford alpaca: An instruction-following llama model. *GitHub repository*, 2023. Available on: https://github.com/tatsu-lab/stanford_alpaca. Cited 2 times in pages 29 and 43.

TAVARES, C. et al. Tiny titans: Efficient large vision, language and multimodal models through pruning. *Conference on Graphics, Patterns and Images (SIBGRAPI) - Tutorial*, 2025. Cited 2 times in pages 18 and 29.

TOUVRON, H. et al. Llama: Open and efficient foundation language models. *ArXiv*, 2023. Available on: https://arxiv.org/abs/2302.13971. Cited 2 times in pages 27 and 41.

TOUVRON, H. et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, 2023. Available on: https://arxiv.org/abs/2307.09288. Cited 2 times in pages 27 and 41.

VASWANI, A. et al. Attention is all you need. *Neural Information Processing Systems (NeurIPS)*, 2017. Cited 5 times in pages 19, 20, 22, 23, and 24.

WAN, Z. et al. Efficient large language models: A survey. *Transactions on Machine Learning Research (TMLR)*, 2023. Cited in page 13.

WANG, C. et al. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. *International Conference on Machine Learning (ICML)*, 2019. Cited in page 13.

WANG, H. et al. Recent advances on neural network pruning at initialization. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022. Cited in page 47.

WANG, H. et al. Neural pruning via growing regularization. *International Conference on Learning Representations (ICLR)*, 2021. Cited 2 times in pages 13 and 14.

WEI, J. et al. Finetuned language models are zero-shot learners. *International Conference on Learning Representations (ICLR)*, 2022. Cited in page 25.

XIA, M. et al. Sheared LLaMA: Accelerating language model pre-training via structured pruning. *International Conference on Learning Representations (ICLR)*, 2024. Cited 2 times in pages 30 and 54.

XIAO, G. et al. Smoothquant: Accurate and efficient post-training quantization for large language models. *International Conference on Machine Learning (ICML)*, 2023. Cited in page 13.

YAMAMOTO, B. L. et al. Compressing llms with mop: Mixture of pruners. *ArXiv*, 2026. Available on: https://arxiv.org/abs/2602.06127. Cited in page 18.

ZELLERS, R. et al. Hellaswag: Can a machine really finish your sentence? *Association for Computational Linguistics (ACL)*, 2019. Cited in page 41.

ZHANG, B.; SENNRICH, R. Root mean square layer normalization. *Neural Information Processing Systems (NeurIPS)*, 2019. Cited 2 times in pages 20 and 24.

ZHANG, Q. et al. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *International Conference on Learning Representations (ICLR)*, 2023. Cited in page 26.

ZHANG, Y. et al. Plug-and-play: An efficient post-training pruning method for large language models. *International Conference on Learning Representations (ICLR)*, 2024. Cited 3 times in pages 32, 39, and 43.

ZHANG, Y. et al. Learning efficient image super-resolution networks via structure-regularized pruning. *International Conference on Learning Representations (ICLR)*, 2022. Cited 2 times in pages 13 and 14.

ZHOU, A. et al. Learning n:m fine-grained structured sparse neural networks from scratch. *International Conference on Learning Representations (ICLR)*, 2021. Cited in page 33.

ZHOU, Y. et al. Large language models are human-level prompt engineers. *International Conference on Learning Representations (ICLR)*, 2023. Cited in page 12.

ZHOU, Y.; YEN, G. G.; YI, Z. Evolutionary shallowing deep neural networks at block levels. *IEEE Trans. Neural Networks Learn. Syst. (IEEE TNNLS)*, 2022. Cited in page 13.

ZHU, X. et al. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics (TACL)*, 2024. Cited in page 13.