Efficient LLMs with AMP: A̲ttention Heads and M̲LP P̲runing

Leandro Giusti Mugnaini*§, Bruno Lopes Yamamoto*§, Lucas Lauton de Alcantara*, Victor Zacarias‡, Edson Bollis†, Lucas Pellicer†, Anna Helena Reali Costa* and Artur Jordao*

* Escola Politécnica, ‡ Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil

† Instituto de Ciência e Tecnologia Itaú (ICTi), São Paulo, Brazil

***Abstract*—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 paper, we introduce AMP: A̲ttention Heads and M̲LP P̲runing, a novel structured pruning method that efficiently compresses LLMs by removing less critical structures within Multi-Head Attention (MHA) and Multilayer Perceptron (MLP). By projecting the input data onto weights, AMP assesses structural importance and overcomes the limitations of existing techniques, which often fall short in flexibility or efficiency. In particular, AMP surpasses the current state-of-the-art on commonsense reasoning tasks by up to 1.49 percentage points, achieving a 30% pruning ratio with minimal impact on zero-shot task performance. Moreover, AMP also improves inference speeds, making it well-suited for deployment in resource-constrained environments. We confirm the flexibility of AMP on different families of LLMs, including LLaMA and Phi.**

***Index Terms*—LLM, Pruning, Structured Pruning, Model Compression, Green AI.**

I. Introduction

Within the evolving landscape of Artificial Intelligence, Large Language Models (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 [1]–[3]. However, this performance often comes at the cost of larger model sizes, with models such as DeepSeek-V3 [4] reaching the mark of 671 billion parameters.

This substantial size introduces significant challenges for the deployment of LLMs in low-resource and time-critical applications due to high computational costs and slow inference [5]. To overcome these challenges, some techniques such as pruning [6] and quantization [7], [8] aim to reduce the model size, thereby lowering computational overhead.

Recent studies confirm pruning as a promising solution to compress models as it maintains predictive ability and is often hardware-agnostic [9]–[13]. Within the field of LLMs, pruning techniques fall into three main categories: structured, semi-structured and unstructured pruning.

Structured pruning removes entire components – such as attention heads or layers – while preserving the overall network structure without introducing sparsity (i.e., without zeroing out a significant portion of the model's parameters) [6]. However, the removal of larger and potentially more critical components may result in performance degradation, typically requiring Parameter-Efficient Fine-Tuning (PEFT) techniques for performance recovery [14]. Due to the removal of complete components, structured pruning usually achieves inference acceleration and memory reduction without the need for specialized hardware or software [15].

Semi-structured (a.k.a. structured $N : M$) pruning promotes model sparsity by removing groups of consecutive parameters following a pruning mask [16]. Specifically, structured $N : M$ sparsity requires that at least N out of every M consecutive weights be non-zero [17], [18]. While this is a promising technique, it requires specialized hardware to achieve practical speedup, making it less suitable for deployment on consumer-grade GPUs [17].

Finally, unstructured pruning removes weights without considering any pattern, increasing model sparsity by zeroing out parameters [10], [16]. However, the resulting irregular sparse matrices often fail to achieve efficiency comparable to dense matrices on general-purpose hardware [11], [18]. To improve efficiency, the resulting architecture requires specific software or hardware optimizations, often resulting in lower inference speedups when compared to other forms of pruning [18].

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 [15], [16]. In this context, a data-driven approach takes weights and input data into account on component evaluation [10]. In contrast, loss-based metrics assess structure importance by measuring their impact on the loss function. Successful methods in this sphere typically employ Taylor expansion [9], [15], [19]. Different techniques also encompass regularization into their methodology [20], [21]. Finally, some approaches consider the similarity between weights or representations, employing metrics such as cosine similarity to determine the least important structures [22].

Despite the effectiveness of the aforementioned strategies, existing pruning methods often face trade-offs involving per-

§Equal contribution.

formance, latency, or the requirement for specialized hardware. Hence, there remains a need for simpler, hardware-agnostic techniques that effectively compress LLMs with minimal performance loss while achieving tangible inference speedups. In this work, we introduce Attention Heads and MLP Pruning (AMP), a pruning technique that addresses gaps in existing methods. Our criterion removes attention heads and MLP neurons across layers to achieve target compression rates. AMP uses the magnitude of activations – rather than solely the weights – to evaluate the importance of structures within each layer, ensuring uniform pruning throughout all layers. Among our contributions, we highlight the following:

- We propose AMP, an efficient and effective structured pruning method that surpasses the state-of-the-art techniques of LLM pruning, providing practical inference speedups without the need for specialized hardware. Notably, our method achieves up to $1.25\times$ inference speedup while surpassing existing structured pruning techniques by up to 1.49 percentage points in average task accuracy.
- In contrast to existing approaches, AMP identifies unimportant structures within minutes and requires a negligible fine-tuning process for performance recovery, making it both practical and resource-efficient.
- 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 method is applicable to different LLM families, requiring minimal modifications. Additionally, AMP is aligned with Green AI principles when deploying LLMs at scale, leading to direct cost savings and reduced CO_2 emissions.

Code and models are available at: https://github.com/c2d-usp/Efficient-LLMs-with-AMP.

II. Related Work

We contrast our method with the three categories of pruning [6], [23]: structured, semi-structured and unstructured.

Structured Pruning. LLM-Pruner [9] locates discrete neuron clusters in LLMs based on their internal dependencies, measures the importance of each one based on loss changes, and prunes those with lower importance. Sheared LLaMA [24] compresses LLMs into compact architectures and continues pre-training with dynamic batch loading, using loss values to guide its pruning and training processes.

Unlike previous works, Shortened LLaMA [15] eliminates entire Transformer layers using Taylor expansion (based on differences in training loss) or a perplexity-driven criterion (based on perplexity changes). SliceGPT [12] prunes models by removing rows and columns according to Principal Component Analysis, achieving strong performance. However, it requires the addition of many extra parameters to maintain compatibility with internal model dimensions.

Similarly to SliceGPT, DISP-LLM [11] eliminates modules by removing rows and columns via selection matrices and employs ReinMax to optimize layer width. In contrast, FLAP [25] multiplies the variance of a calibration dataset by the squared weights norm to score channels, standardizes these scores, removes channels accordingly, and adjusts bias. As a result, FLAP not only depends heavily on the calibration dataset but also introduces bias into the model. According to Xia et al. [24], the varying layer widths introduced by these methods lead to additional training and inference overhead.

Alternatively, PruneNet [26] employs a policy learning-based compression for LLMs. Essentially, policy gradient trains a learner model penalizing it with Kolmogorov-Smirnov distance between singular value distributions for pruned and unpruned matrices. Despite an innovative approach, the study leaves pruning mixed attention heads and MLPs unaddressed, a field we explore through AMP. LLM Surgeon [13] provides a comprehensive framework for unstructured, semi-structured, and structured pruning, employing Kronecker-factored curvature approximations of the target loss landscape for comprehensive pruning. While the method achieves competitive results, it requires significant computational resources.

In contrast to the existing approaches we mentioned above, our AMP strategy prunes attention heads and MLP neurons by using activation magnitudes from a small number of forward passes, rather than relying on computationally expensive loss-based or projection-driven heuristics. Our effective design avoids rigid constraints, such as pruning only the Multilayer Perceptron. By applying uniform pruning across layers, AMP also prevents the creation of layers with varying widths, thereby avoiding additional overhead.

Semi-Structured Pruning. COPAL [27] prunes weights in Large Language Models by analyzing directional derivatives of the loss function across datasets. Although effective, it requires numerous differential operations to determine importance, thereby increasing its computational cost. Wanda [10] identifies sparse sub-networks in LLMs by ranking each weight using the product of its absolute value and its squared input feature norm. However, as Gao et al. [11] argue, Wanda excels only as an unstructured pruning technique, a form of sparsity that does not lead to inference speedups in the resulting model. Unlike these methods, AMP does not require specialized hardware to achieve practical speedups. Furthermore, while existing techniques are constrained to fixed sparsity patterns such as 2:4 or 4:8, limiting their compression rates, AMP offers greater flexibility in achieving desired compression levels without being bound by predefined sparsity structures.

Unstructured Pruning. SparseGPT [16] formulates pruning as a sparse regression solved via an efficient approximate mechanism, achieving up to 60% sparsity while preserving model performance without traditional retraining. However, it performs less accurately on small to medium-sized models compared to larger networks. Plug-and-Play [28] introduces a one-shot post-training pruning combining Relative Importance and Activation (RIA) with Channel Permutation. Despite avoiding retraining, it exhibits sensitivity to calibration data and increases complexity, particularly for larger models. Moreover, these methods often fail to provide practical inference speedups, as modern GPUs accelerate inference for semi-structured patterns (e.g., 2:4) [17], [18]. On the other hand, AMP follows a structured approach and achieves tangible inference speedups using a lightweight, efficient procedure.

III. Proposed AMP method

Residual connections link the input of different components of Large Language Models to its output, creating a continuous flow of information through the network. This path of information is commonly referred to as the residual stream [29].

Building on this concept, our method measures the importance of each component based on its addition to the residual stream, making it possible to prune components with lower contribution. In particular, we prune attention heads from Multi-Head Attention (MHA) and neurons from Multilayer Perceptron (MLP) within each layer of the model. This approach provides a more faithful representation of how the network internally processes information than simple weight-only or activation-only heuristics.

The idea behind our method is simple; however, due to the sophisticated architecture of Transformers, we start by defining notations and explaining the two main components involving our AMP criterion: Multi-Head Attention (MHA) and Multilayer Perceptron (MLP). Although the description and explanation focus on the LLaMA-2 7B model, the method is applicable to other Transformer-based LLMs with similar architecture, as we shall see in the experiments.

A. Multi-Head Attention

Each Multi-Head Attention (MHA) mechanism involves several attention heads [30]. As these are independent units, they process the input data in parallel. MHA then adds the combined output to the residual stream.

In each attention component of the Transformer architecture, MHA comprises exactly N attention heads. Consider S the sequence length and d_{model} the hidden size dimension. Given an input $\mathbf{X} \in \mathbb{R}^{\mathrm{S} \times d_{\mathrm{model}}}$, each attention head n within MHA performs projections into the Key ($\mathbf{K}$), Query ($\mathbf{Q}$) and Value ($\mathbf{V}$) matrices as follows:

$$\mathbf{K}_n = \mathbf{X}\mathbf{W}_n^K, \mathbf{Q}_n = \mathbf{X}\mathbf{W}_n^Q, \mathbf{V}_n = \mathbf{X}\mathbf{W}_n^V, \tag{1}$$

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.

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

$$\text{Attention}(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2}$$

Then, for a given layer, we express the n-th attention head as:

$$\mathbf{h}_n = \text{Attention}(\mathbf{K}_n, \mathbf{Q}_n, \mathbf{V}_n). \tag{3}$$

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

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N)\mathbf{W}_O. \tag{4}$$

The LLaMA-2 7B model has a single MHA module in each layer [31], which integrates 32 attention heads (i.e., N=32).

B. Multilayer Perceptron

Each LLaMA layer contains a Multilayer Perceptron (MLP) component that uses the SwiGLU mechanism. This component consists of three linear projections: Up, Gate, and Down Projections. Let $x \in \mathbb{R}^{S \times d_{model}}$ be the input tensor. The SwiGLU MLP is then the following:

$$\text{MLP}(x) = (\text{SiLU}(x\mathbf{W}_{Gate}) \odot (x\mathbf{W}_{Up}))\mathbf{W}_{Down}, \tag{5}$$

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

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 $\mathbb{R}^{d_{model}}$ by $\mathbf{W}_{Down}$.

C. Proposed AMP method

We define our AMP importance criterion as a combination of two independent contributions: the MHA component and the MLP component.

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 performance 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 $\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$ 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 [32] – 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 the MHA mechanism.

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 4).

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:

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

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

Then, the output of MHA becomes

$$\mathbf{O} = \mathbf{H}\mathbf{W}_o = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_N \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \tag{6}$$

and in a summation form

$$\mathbf{O} = \mathbf{h}_1\mathbf{W}_1 + \mathbf{h}_2\mathbf{W}_2 + \cdots + \mathbf{h}_N\mathbf{W}_N = \sum_{n=1}^{N} \mathbf{h}_n\mathbf{W}_n. \tag{7}$$

Equation 7 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.

Finally, we define the importance $\mathbf{I}_n$ of each head as the ℓ_1 norm of $\mathbf{h}_n\mathbf{W}_n$

$$\mathbf{I}_n = \|\mathbf{h}_n\mathbf{W}_n\|_1 . \tag{8}$$

Our importance criterion given by Equation 8 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, by projecting the data onto the weights, differing 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.

AMP – MLP Component. We propose pruning neurons from the up and gate projections in the SwiGLU MLP (see Equation 5); thereby reducing the MLP's intermediate size (d_i) while maintaining the hidden dimension (d). The SwiGLU mechanism employs element-wise multiplication between the up and gate outputs, so the neurons must be pruned in pairs from both projections. To identify which pairs to prune, the MLP component of AMP accounts for both weight parameters and input magnitudes. In contrast to simple ℓ_1 or ℓ_2 norms of the weights – whose scores could be sometimes misleading as they ignore the scale of the input [16] – the MLP component measures 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:

$$\mathbf{I}_m = \frac{1}{S}\sum_{s=1}^{S} \left|\text{SiLU}\left((x_s\mathbf{W}_{Gate})_m\right) \cdot (x_s\mathbf{W}_{Up})_m\right|, \tag{9}$$

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 $x_s\mathbf{W}_{Up}$.

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 formalism, 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 [9], [10], [15], [28], 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.

$AMP(\cdot, \mathbf{X}_{sub})$ is the AMP method that extracts the mean feature representations (activations) from MLP neurons and attention heads, according to Equations 8 and 9, for each layer ℓ using the samples in $\mathbf{X}_{sub}$.

We determine the final importance of each component using the activation values. To achieve the desired compression rate c, 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. Algorithm 1 summarizes the pruning process with AMP.

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

IV. Experiments

A. Experimental Settings

Large Language Models. In order to compare our method against existing pruning techniques, we consider four open-source LLMs commonly used in pruning research: LLaMA 7B [33], LLaMA-2 7B [31], Phi-1.5 and Phi-2 [34].

Evaluation and Datasets. We evaluate the mean performance of each model at different pruning ratios. For a fair comparison, we opt for benchmarks of previous works [9], [11]–[13], [15]. We use the EleutherAI LM Harness framework [35] to perform zero-shot task classification on common sense reasoning datasets: WinoGrande [36], HellaSwag [37], ARC-e / ARC-c [38] and PIQA [39]. Following Ouderaa et al. [13], we report the normalized accuracy for all benchmarks, except for WinoGrande, where we report accuracy. Additionally, we perform a perplexity (PPL) analysis on WikiText2 [40], using the procedures by Ashkboos et al. [12].

Inference Speedup Evaluation. To evaluate inference speedup, we adopt the definition of latency from Sheng et al. [5]. For a batch size J and an output sequence length K, latency T is defined as the time required to generate JK output tokens based on the input prompts. We measure speedup as the relative difference in latency between the pruned model and its original, unpruned, counterpart. Following Kim et al. [15], 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.

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 [41]. This choice aligns with prior studies that consider a small calibration set (typically ranging from 10 to 128 samples) to conduct the pruning process [9], [10], [15], [28]. Following Ma et al. [9], we employ a post-training process using low-rank adaptation, LoRA [42], to recover the model performance. We set the 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). It is worth mentioning that this setup follows previous work [9] and ensures the benefits of our method do not come from hyperparameter customizations.

B. 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 I presents 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 [12] and DISP-LLM [11], using a more efficient approach.

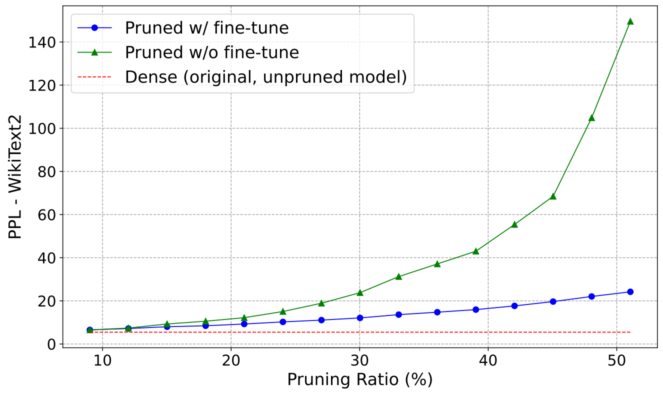

Fig. 1: Pruning impact of AMP on WikiText2 perplexity for LLaMA-2 7B. The red dashed horizontal line indicates the performance of the original model, while the blue and green lines represent the performance of the pruned model with and without fine-tuning, respectively.

Although AMP delivers comparable performance for the LLaMA family, Phi models exhibit a more pronounced drop. Since smaller models generally have less capacity than larger ones, they may be more sensitive to pruning, leading to greater performance declines [16].

Figure 1 shows PPL results on the WikiText-2 dataset. As lower PPL indicates better performance, 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 51.06% pruning ratio, the no-fine-tuning approach exhibits a PPL 6.18 times higher than the former.

As a final note, Kim et al. [15] 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.

C. Inference Speedup

Even though the number of parameters offers theoretical insights about performance, it may fail to fully capture the model efficiency when used as a standalone metric [43]. From 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 [1], the mean latency across runs for

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

TABLE I: Comparison of state-of-the-art structured pruning methods across different LLMs.

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 [9] (NeurIPS, 2023)	61.33	65.34	59.18	37.12	75.57	59.71
	LLM Pruner (+ fine-tuning) [9] (NeurIPS, 2023)	65.11	68.11	63.43	37.88	76.44	62.19
	Shortened LLaMA [15] (ICLR, 2024) (‡)	**68.82**	69.82	64.06	39.93	74.65	63.46
	DISP-LLM [11] (NeurIPS, 2024)	64.72	68.39	64.81	37.12	76.66	62.34
	PruneNet [26] (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 [12] (ICLR, 2024)	61.33	49.62	51.77	31.23	63.55	51.50
	LLM Surgeon [13] (ICLR, 2024)	61.09	60.72	63.09	36.69	73.56	59.03
	Shortened LLaMA [15] (ICLR, 2024) (‡)	61.09	54.97	45.96	34.81	60.99	51.56
	DISP-LLM [11] (NeurIPS, 2024)	**63.93**	62.87	60.1	37.03	73.72	59.53
	PruneNet [26] (ICLR, 2025)	61.09	58.30	53.20	32.94	71.11	55.33
	PruneNet (+ fine-tuning) [26] (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 [12] (ICLR, 2024)	**64.96**	42.54	53.66	31.91	65.45	51.70
	DISP-LLM [11] (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 [12] (ICLR, 2024)	63.14	47.56	53.03	30.29	65.94	51.99
	DISP-LLM [11] (NeurIPS, 2024)	65.19	54.43	63.59	38.48	**73.34**	59.01
	PruneNet [26] (ICLR, 2025)	**67.48**	56.80	**67.55**	**40.61**	72.80	**61.05**
	PruneNet (+ fine-tuning) [26] (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

(‡) 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. In addition, since the authors do not present results for the LLaMA-2 7B model, we run their method on our own.

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 [15] and rivalize with semi-structured approaches like Wanda [10]. The former achieves a 1.13× 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 by leveraging sparse tensor acceleration, thus requiring specialized hardware.

D. Coherence Check of Proposed AMP

To confirm the effectiveness of our method in selecting the least important structures for removal, we propose an experiment 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, 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 [44], [45]. In this experiment, we use the LLaMA-2 7B model at three compression rates across different benchmarks.

Table II 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 across all pruning ratios compared to the original ranking. Interestingly, random pruning, although less effective than our original method, still achieves better performance than the reversed-importance approach. In summary, this contrast reinforces that AMP successfully differentiates between essential and non-essential structures, validating the strategy of selecting lower-importance elements for pruning.

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.

E. 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

TABLE II: 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

TABLE III: Influence of removing isolated components and all at once, using LLaMA-2 7B with a 30% compression rate. The results emphasize that AMP, which combines the pruning of 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 w/ MHA + MLP Components	61.25	65.47	64.31	39.85	74.21	**61.02**
AMP w/ MLP Component	57.14	60.64	59.81	34.64	72.42	56.93
AMP w/ MHA Component	49.49	28.13	30.60	24.23	55.71	37.63

the full AMP criterion that eliminates both components. We follow the same setup as in Subsection IV-B.

Table III 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 percentage points (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, which constitute approximately 67% of the parameters within a layer, provides a more effective reduction in model size compared to pruning attention heads alone. 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. We plan to further investigate the use of different pruning ratios for attention heads and MLP neurons in future work.

F. 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 [46]–[48]. In particular, LLMs are well-known for their high computational costs, both in training and deployment. Hence, they require increasingly large computational resources due to the scaling laws governing their development [49].

In line with Green AI principles, our AMP method achieves up to 1.25× inference speedup, representing a 17.39% direct decrease in operational costs. Furthermore, we achieve a reduction of approximately 19.97% in carbon emissions[2]. 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.

[2]For reproducibility, we estimate these values using the Machine Learning Impact calculator [50].

G. Impact on the Number of Fine-tuning Epochs

To assess the impact of training duration on 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 IV-B.

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%. From these results, we observe that using more than one epoch of fine-tuning does not significantly improve model performance. Therefore, we believe a more promising approach would be to train on much larger datasets instead of using more epochs, since Alpaca contains only 52k samples. However, we opt to use two epochs to create a fair comparison with other works, such as LLM-Pruner [9].

V. Conclusions

We introduce AMP, a novel structured pruning method designed to efficiently identify and eliminate less critical components (attention heads and MLP neurons) from Large Language Models (LLMs), producing highly optimized models. Moreover, our coherence check confirms that AMP correctly ranks the importance of each component. Particularly, when our method removes components ranked as more critical (those it should preserve), the model performance drastically decreases. In addition, our method is computationally efficient, requiring only a few minutes on a consumer-grade GPU (NVIDIA RTX 3090) to identify the components for removal.

Through extensive experiments, we show that the proposed method outperforms state-of-the-art structured pruning approaches. Specifically, AMP surpasses existing techniques by a large margin, achieving mean accuracy improvements of up to 4.81 pp for LLaMA 7B and 9.52 pp for LLaMA-2 7B. Furthermore, perplexity evaluation on WikiText2 corroborates the results by demonstrating the consistency of AMP across many compression rates. Notably, AMP achieves up to 1.25× inference speedup without requiring specialized hardware, further enhancing efficiency and sustainability. Finally, apart from these benefits, our method aligns with Green AI principles, emphasizing the role of AMP in reducing the computational costs of LLMs.

Last but not least, we present evidence of the significance of the post-training stage in restoring model performance after pruning, corroborating previous work [24]. In this context, we hypothesize that extending post-training on larger datasets would yield even greater performance, a direction we plan to explore in future work.

VI. Acknowledgments

The authors would like to thank Instituto de Ciência e Tecnologia Itaú (ICTi) and Programa 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. The authors would like to thank grant #402734/2023-8, National Council for Scientific and Technological Development (CNPq). Artur Jordao Lima Correia would like to thank Edital Programa de Apoio a Novos Docentes 2023. Processo USP nº: 22.1.09345.01.2. Anna H. Reali Costa would like to thank grant #312360/2023-1 CNPq.

References

[1] J. A. et al., "Gpt-4 technical report," 2024.

[2] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in *International Conference on Learning Representations (ICLR)*, 2023.

[3] T. B. et al., "Language models are few-shot learners," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[4] A. L. et al, "Deepseek-v3 technical report," *ArXiv*, 2024.

[5] Y. S. et al., "Flexgen: High-throughput generative inference of large language models with a single GPU," in *International Conference on Machine Learning (ICML)*, 2023.

[6] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, "A survey on model compression for large language models," in *Transactions of the Association for Computational Linguistics (TACL)*, 2024.

[7] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "OPTQ: Accurate quantization for generative pre-trained transformers," in *International Conference on Learning Representations (ICLR)*, 2023.

[8] R. J. et al., "A comprehensive evaluation of quantization strategies for large language models," in *Association for Computational Linguistics (ACL)*, 2024.

[9] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," in *Neural Information Processing Systems (NeurIPS)*, 2023.

[10] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[11] S. Gao, C.-H. Lin, T. Hua, Z. Tang, Y. Shen, H. Jin, and Y.-C. Hsu, "DISP-LLM: Dimension-independent structural pruning for large language models," in *Neural Information Processing Systems (NeurIPS)*, 2024.

[12] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman, "SliceGPT: Compress large language models by deleting rows and columns," in *International Conference on Learning Representations (ICLR)*, 2024.

[13] T. F. A. van der Ouderaa, M. Nagel, M. V. Baalen, and T. Blankevoort, "The LLM surgeon," in *International Conference on Learning Representations (ICLR)*, 2024.

[14] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *Transactions on Machine Learning Research (TMLR)*, 2024.

[15] B.-K. K. et al., "Shortened llama: A simple depth pruning for large language models," in *International Conference on Learning Representations (ICLR) - Workshop*, 2024.

[16] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," in *International Conference on Machine Learning (ICML)*, 2023.

[17] A. Z. et al., "Learning n:m fine-grained structured sparse neural networks from scratch," in *International Conference on Learning Representations (ICLR)*, 2021.

[18] A. M. et al., "Accelerating sparse deep neural networks," *ArXiv*, 2021.

[19] C. Wang, R. Grosse, S. Fidler, and G. Zhang, "Eigendamage: Structured pruning in the kronecker-factored eigenbasis," in *International Conference on Machine Learning (ICML)*, 2019.

[20] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," in *International Conference on Learning (ICLR)*, 2021.

[21] Y. Zhang, H. Wang, C. Qin, and Y. Fu, "Learning efficient image super-resolution networks via structure-regularized pruning," in *International Conference on Learning Representations (ICLR)*, 2022.

[22] A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. A. Roberts, "The unreasonable ineffectiveness of the deeper layers," in *International Conference on Learning Representations (ICLR)*, 2025.

[23] Z. W. et al., "Efficient large language models: A survey," in *Transactions on Machine Learning Research (TMLR)*, 2023.

[24] M. Xia, T. Gao, Z. Zeng, and D. Chen, "Sheared LLaMA: Accelerating language model pre-training via structured pruning," in *International Conference on Learning Representations (ICLR)*, 2024.

[25] Y. An, X. Zhao, T. Yu, M. Tang, and J. Wang, "Fluctuation-based adaptive structured pruning for large language models," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[26] Anonymous, "You only prune once: Designing calibration-free model compression with policy learning," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://openreview.net/forum?id=5RZoYIT3u6

[27] S. Malla, J. H. Choi, and C. Choi, "COPAL: Continual pruning in large language generative models," in *International Conference on Machine Learning (ICML)*, 2024.

[28] Y. Z. et al., "Plug-and-play: An efficient post-training pruning method for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[29] N. E. et al., "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021.

[30] A. V. et al., "Attention is all you need," in *Neural Information Processing Systems (NeurIPS)*, 2017.

[31] H. T. et al., "Llama 2: Open foundation and fine-tuned chat models," *ArXiv*, 2023.

[32] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2024.

[33] H. T. et al., "Llama: Open and efficient foundation language models," *ArXiv*, 2023.

[34] Y. Li, S. Bubeck, R. Eldan, A. D. Giorno, S. Gunasekar, and Y. T. Lee, "Textbooks are all you need ii: phi-1.5 technical report," *ArXiv*, 2023.

[35] L. G. et al., "A framework for few-shot language model evaluation," *Zenodo*, 2024.

[36] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *ArXiv*, 2019.

[37] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" in *Association for Computational Linguistics (ACL)*, 2019.

[38] P. C. et al., "Think you have solved question answering? try arc, the ai2 reasoning challenge," *ArXiv*, 2018.

[39] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[40] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *ArXiv*, 2016.

[41] R. T. et al., "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[42] E. J. H. et al., "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[43] M. Dehghani, Y. Tay, A. Arnab, L. Beyer, and A. Vaswani, "The efficiency misnomer," in *International Conference on Learning Representations (ICLR)*, 2022.

[44] H. Wang, C. Qin, Y. Bai, Y. Zhang, and Y. Fu, "Recent advances on neural network pruning at initialization," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[45] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. V. Gool, "Revisiting random channel pruning for neural network compression," in *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[46] R. S. et al., "Green AI," in *Association for Computing Machinery (ACM)*, 2020.

[47] A. F. et al., "Llmcarbon: Modeling the end-to-end carbon footprint of large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[48] Anonymous, "Holistically evaluating the environmental impact of creating language models," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://openreview.net/forum?id=04qx93Viwj

[49] J. H. et al., "Training compute-optimal large language models," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[50] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," in *Neural Information Processing Systems (NeurIPS)*, 2019.