Tiny Titans: Efficient Large Vision, Language and Multimodal Models through Pruning

Carolina Tavares[†], Leandro Mugnaini[†], Gustavo Henrique do Nascimento[†], Ian Pons[†], Keith Ogawa[†],
Guilherme Stern[†], Lucas Libanio[†], Aline Paes[‡], Anna Helena Reali Costa[†] and Artur Jordao[†]
[‡]Universidade Federal Fluminense, [†]Universidade de Sao Paulo, Brazil

***Abstract*—Notable progress in solving complex reasoning tasks relies on large models. Unfortunately, developing these models demands substantial computational resources and energy consumption. Hence, the industry pushes the most significant advances in state-of-the-art models and draws the attention of the scientific community to the environmental impact of AI (GreenAI). Pruning emerges as an effective mechanism to address the capacity-computational cost dilemma by eliminating structures (weights, neurons or layers) from deep models. This tutorial introduces theoretical and technical foundations within this promising, active and exciting field. It delves into pruning techniques as a pillar of GreenAI and a foundation for the next wave of efficient large vision, language, and multimodal models. Our tutorial also covers how existing forms of pruning impact efficiency gains, guiding participants to make informed choices for their scenario and infrastructure. Specifically, we equip participants with the basics and key recipes to effectively apply pruning in practical computer vision scenarios. Additional material is available at: github.com/arturjordao/TinyTitans**

I. Introduction

Deep learning drives the most notable progress in cognitive tasks such as visual and mathematical reasoning, and now shapes various aspects of our modern society [1], [2]. On the other hand, developing high-capacity and state-of-the-art models requires massive computing infrastructure [3]. For example, training the Gemini Ultra model costs around USD 191,400,000; hence, the industry pushes the most significant advances in frontier AI research [1].

Another important aspect behind computing demand is the carbon emissions resulting from energy consumption. Studies estimate that carbon footprints of training large models are comparable to those of entire industrial sectors, such as aviation [3], [4]. In particular, AI accounts for 0.1–0.28% of global greenhouse gas emissions [2]. Even relatively small foundation models emit as much carbon as burning 1.5 gasoline tanker trucks [3]. Interestingly, while studies often associate these values with Large Language Models (LLMs), image generation ranks highest in carbon footprint [1]. Previous factors motivate intense research into more efficient and sustainable models and draw the attention of the scientific community to the environmental impact of AI—GreenAI.

The capacity-computational cost dilemma naturally calls for strategies to reduce computational overhead while preserving learning abilities. Among existing solutions to overcome this, pruning emerges as an effective and hardware-agnostic technique for improving model efficiency [5], [6]. Concretely, state-of-the-art pruning techniques reduce computing costs by more than 75% with only a marginal drop in generalization [5]. The central idea behind this family of techniques is to remove structures from the architecture, such as weights, filters or even entire layers. Figure 1 illustrates existing pruning strategies.

979-8-3315-8951-6/25/$31.00 ©2025 IEEE

In this tutorial, we explore topics related to GreenAI and provide a comprehensive overview of pruning techniques. Our goal is to equip participants with the theoretical background and technical tools essential for applying pruning in practical scenarios. Specifically, we cover. (i) The environmental implications of developing high-capacity, state-of-the-art models and introduce the concept of GreenAI. (ii) An in-depth review of modern and successful pruning strategies, highlighting potential avenues for development for each one. (iii) Emerging trends and open challenges that could shape the next generation of compression techniques. (iv) Practical demonstrations of pruning in popular computer vision models. Since we believe future progress in general-purpose AI depends on the synergy between vision and language models, part of this tutorial also includes discussions and practical demonstrations of pruning in popular Large Language Models.

After participating in this tutorial, attendees will be familiar with: (i) The potential of pruning to reduce the computational demand of computer vision models. (ii) The compromise between computational gains and predictive ability in computer vision models. (iii) GreenAI-related metrics.

To support this learning experience, we provide an open-source pruning framework designed for computer vision models, including standard architectures such as Residual Networks. In addition, we provide a framework for pruning widely used LLMs, developed as part of our tutorial. We will release all code and materials from the tutorial online to encourage further engagement from the research community.

II. General Pruning Framework

The strategy of pruning consists of removing structures from deep networks. Its success relies on two factors: overparameterization and plasticity of deep models. In the overparameterized regime, networks have more capacity than required to solve the problem at hand; hence, some structures become redundant and, thus, unimportant. Plasticity, on the other hand, refers to the ability of neural networks to recover from damage to their structure [7]. This concept aligns with pruning as it allows recovering the predictive ability of deep models after removing certain structures.

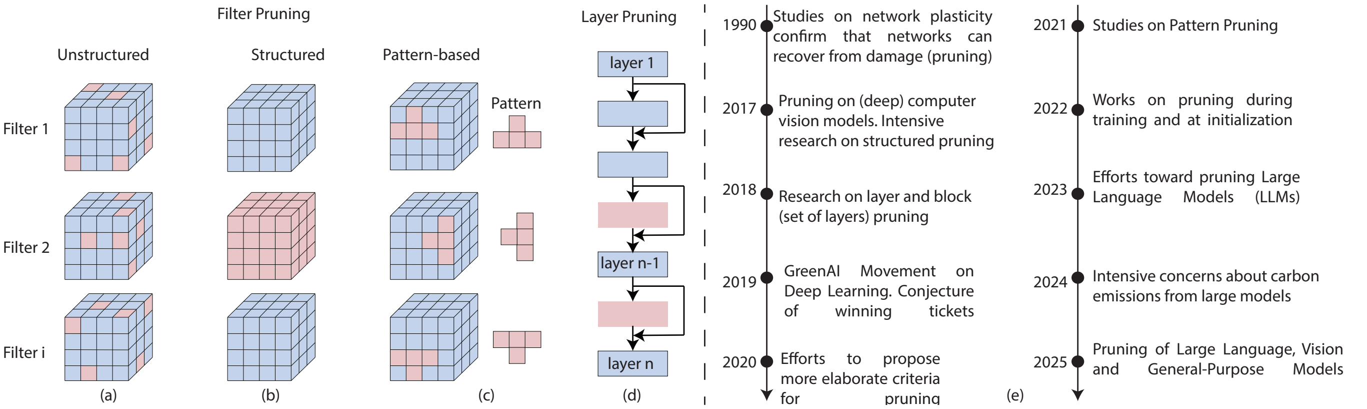
Filter Pruning
Unstructured
Structured
Pattern-based
Pattern
Filter 1
Filter 2
Filter i
(a)
(b)
(c)
Layer Pruning
layer 1
layer n-1
layer n
(d)
1990 Studies on network plasticity confirm that networks can recover from damage (pruning)
2017 Pruning on (deep) computer vision models. Intensive research on structured pruning
2018 Research on layer and block (set of layers) pruning
2019 GreenAI Movement on Deep Learning. Conjecture of winning tickets
2020 Efforts to propose more elaborate criteria for pruning
2021 Studies on Pattern Pruning
2022 Works on pruning during training and at initialization
2023 Efforts toward pruning Large Language Models (LLMs)
2024 Intensive concerns about carbon emissions from large models
2025 Pruning of Large Language, Vision and General-Purpose Models
(e)

Fig. 1: Different families of pruning techniques. Blue and red areas indicate, respectively, unpruned and pruned structures (weights, filters or layers). Each cuboid (tensor) in (a)-(c) stands for a $k \times k$ filter. (a) Unstructured pruning removes weights at arbitrary locations, thus yielding sparse tensors that require specialized frameworks or modern hardware support to achieve practical improvements. (b) Structured pruning eliminates entire structures, enabling computational gains in a hardware-agnostic manner. (c) Pattern-based pruning removes structured patterns (i.e., a fixed number of locally connected weights) within filters, offering practical gains on modern GPUs. (d) A form of structured pruning that removes entire layers (more precisely blocks of layers) and promotes notable speed-ups in model latency. (e) Noteworthy progress and seminal research in pruning.

Leveraging the previous factors, a typical pruning algorithm involves the following steps. **First**, the algorithm defines the types of structures to consider for removal. Early studies focused solely on the removal of weights [8], [9] (Figure 1 (a)), while modern approaches eliminate large structures, such as neurons (i.e., filters in convolutional networks) or layers [5], [10]–[14] – see Figure 1 (b) and (d). Following a taxonomy based on the structure that pruning removes, most strategies fall into three main categories: unstructured, structured, and pattern-based. Such categorization directly impacts the level of sparsity (i.e., compression or pruning ratio) and hardware efficiency achieved, as we discuss in Sections III, IV and V.

The **second step** establishes a pruning criterion to identify the structures for removal. This criterion plays a crucial role, since it defines how pruning decisions occur and directly affects the balance between computational efficiency and predictive capacity. Section VI introduces an in-depth discussion of existing criteria and their impact on pruning effectiveness.

In the **third step**, the algorithm adjusts the weights of the survival structures to compensate for the changes in the architecture and, hence, in the predictive ability. This phase ensures the pruned network recovers from structural changes and adapts the parameters to the new architecture. Section VII provides an overview of successful fine-tuning strategies within the scope of pruning.

In computer vision, top-performing pruning algorithms repeat the previous process K times iteratively [15]–[17]. Another common variation involves performing a one-shot pruning, where the entire procedure occurs in a single step.

Algorithm 1 summarizes the steps of the pruning process and refers to the sections that delve deeper into the relevant aspects. Overall, in each iteration of Algorithm 1, a pruning criterion c assesses the importance S of the structures in the network $\mathcal{F}$. By leveraging this importance, the algorithm removes $p\%$ of the least important structures, resulting in a new network $\mathcal{F}'$. Then, the algorithm adjusts the parameters and repeats the process until completing all K iterations. The result of a pruning iteration is a pruned model ready for inference or other applications. Throughout the text, we use the terms *pruned model* and *subnetwork* interchangeably.

Algorithm 1 Pruning Structures of Deep Networks

```
Input: Network $\mathcal{F}$, Pruning Criterion $c$, Pruning Ratio $p$
Input: Number of Iterations $K$
Output: Pruned Network $\mathcal{F}'$ (a.k.a Subnetwork)
for $k \leftarrow 1$ to $K$ do
    $S \leftarrow c(\mathcal{F})$ ▷ Set importance for each structure▷Sec. VI
    $I \leftarrow p\%$ Unimportant structures based on $S$
    $\mathcal{F}' \leftarrow \mathcal{F} \setminus I$ ▷ Removes the structures indexed by $I$
    Update $\mathcal{F}'$ ▷ Sec. VII
    $\mathcal{F} \leftarrow \mathcal{F}'$
end for
```

While the description above covers a general pruning algorithm, existing efforts focus on variations and improvements at each step. In the following, we explore these advancements in detail, examining how researchers tailored them to address specific challenges and improve the efficiency and viability of deep models, particularly in resource-constrained systems.

III. Unstructured Pruning

Unstructured pruning is a fine-grained approach to reducing the computational complexity of deep models. It typically involves masking certain weights to zero. Because neurons calculate their output using a dot-product operation, masking some weights to zero effectively removes the influence of those specific connections. As a result, *zeroed-out* weights do not contribute to the final output of the neuron [18].

In the pruning literature [5], [19], this mechanism is referred to as unstructured pruning because it removes weights at arbitrary locations (see Figure 1 (a)), leading to sparse tensors without a regular pattern. This technique leaves the network with fewer active weights, resulting in sparse matrix computations. Although simple and effective, such sparsity creates a gap between theoretical and practical computational improvements because existing deep learning frameworks and hardware are not fully optimized for this type of computation [20]. However, recent NVIDIA GPUs, including the A100 and the more recent Hopper architecture [21], target sparse computing and improve its efficiency.

One of the pioneering works in unstructured pruning is the work by Han et al. [18], who focus on removing weights with low ℓ_1-norms. Although LeCun et al. [8] and Hassibi et al. [22] had previously explored unstructured pruning, Han et al. [18] popularized and advanced the study of unstructured pruning in over-parameterized models such as VGG. Particularly, their approach highlights the underlying properties of these models, thereby helping to advance the field.

Although simple, Sun et al. [23] noted that ℓ_p-norm criteria may fail to identify unimportant weights when the input scale varies significantly. As a solution, they proposed combining weight magnitudes and input activations to eliminate unimportant weights without the need for retraining. Recent studies focus on designing better criteria for unstructured pruning. For example, Frantar et al. [24] use a sparse regression solver to remove weights based on Hessian row reconstruction. Their approach demonstrates promising results by obtaining at least 50% sparsity in one-shot without any retraining.

One significant concept involving unstructured pruning is the *Lottery Ticket Hypothesis* [25]. It posits that within large, dense networks, there exist sparse subnetworks – or *tickets* – that, when trained from the same initialization, achieve comparable accuracy to the original network. Subnetworks that satisfy this are named *winning tickets*. Important advances arise from the LTH, such as sampling winning tickets during or even before the training process [26], [27].

Takeaways. Unstructured pruning offers a simple approach to achieve higher compression rates without sacrificing accuracy. In particular, it can compress a ResNet50 by $5.96\times$ with no accuracy loss [28]. However, these gains rarely translate into speedups on general-purpose GPUs, since most tools do not automatically skip zeroed weights. Sparsity also requires careful memory and processing management [20], [29].

IV. Structured Pruning

Structured pruning focuses on removing entire structures, such as neurons, filters, or layers. Unlike unstructured pruning, structured pruning maintains dense operations by reducing the tensor size directly (see Figure 1 (b)). Importantly, its structured nature enhances compatibility with existing hardware and software, facilitating practical acceleration improvements.

To the best of our knowledge, Li et al. [30] conducted the first successful structured pruning by extending the ℓ_1-norm criterion by Han et al. [18] to prune filters in convolutional networks rather than individual weights.

Existing literature categorizes structured pruning according to the type of structure it targets [5], [19]: filters or entire layers. Within the first category, existing studies treat filters and channels interchangeably [5], [19].

Building on the foundational work by Li et al. [30], several researchers refined structured filter pruning by developing adaptive methods that guide pruning decisions more effectively [31]–[33]. For example, He et al. [31] proposed adjusting pruning parameters layer by layer to better balance compression and accuracy. More recently, Gupta et al. [34] proposed a torque-based criterion inspired by physics. Specifically, the authors adapt the concept of torque, a rotational force from physics, to quantify filter importance.

A different form of structured pruning involves removing entire layers or blocks of layers from a neural network. For consistency and clarity, we opt to use the term *layer pruning* rather than *block pruning*. Terminology aside, this strategy leads not only to reductions in model size and computational cost, but also to notable improvements in inference latency. Unlike filter pruning, often aimed at reducing Floating Point Operations (FLOPs) or parameter count while maintaining model depth, layer pruning shortens the execution path by reducing the number of sequential operations. The key to the success of layer pruning is the fact that information in deep networks propagates through multiple paths; therefore, layers do not always strongly depend on each other [6], [35].

The potential of layer pruning depends on the network architecture. It turns out that models with multiple information paths, such as ResNets and Transformers, tolerate layer removal better than traditional plain networks (i.e., VGG) [35]. Within this sphere of research, some works suggest that pruning blocks of layers is more effective than removing individual layers [36], [37]. This strategy reduces the risk of disrupting the overall functionality of the network, as blocks often operate semi-independently. In this direction, Zhang et al. [38] showed that reinitializing specific layers leads to significant accuracy drops, reinforcing the need for careful selection of layers to prune. To support this selection, previous works propose criteria that capture the role of large structures more effectively than traditional weight-based scores [36], [37], [39]. Also, Dror et al. [40] and Fu et al. [41] employed structural reparameterization to merge layers, simplifying network depth and mitigating the issue of low variance in weight-based scores.

Takeaways. Structured pruning improves model efficiency and maintains hardware compatibility. High compression rates often lead to greater accuracy degradation compared to unstructured counterpart, since they remove entire structures [28], [42]. Structured layer pruning is particularly attractive because it shortens the execution path by reducing the number of sequential operations, thereby significantly speeding up latency.

V. Pattern Pruning

Pattern pruning (a.k.a semi-structured pruning) bridges the gap between unstructured and structured pruning: it provides high compression rates with minimal accuracy loss and is still hardware-friendly. This approach removes structured patterns (i.e., locally connected weights) from kernels/tensors [28], [42], [43] (see Figure 1 (c)). While unstructured pruning removes individual weights without constraints—resulting in irregular sparsity—pattern pruning enforces regular structures that enable efficient parallel execution and performance gains. For example, grouping and executing similar patterns consecutively enables compiler optimizations for memory access and minimizes redundant loads [19], [29].

Modern pattern pruning strategies adopt an N:M pruning mask (e.g., 2:4 or 4:8) [13], [23]. Specifically, N:M sparsity forces N out of every M consecutive weights to remain nonzero, leveraging advanced GPUs to achieve speed-ups [28].

While N:M sparsity is a prevalent approach, alternative pattern-based methods exist. For example, Ma et al. [43] introduced a pool of patterns derived from image theory and applied convex optimization to assign them to different kernels. Their work showed that pattern pruning also enhances feature map quality by encouraging image smoothing and reducing noise, leading to sharper edges and textures.

Takeaways. Pattern pruning offers a promising trade-off between compression rate and accuracy. By enforcing regular sparsity patterns, it enables speed-ups on modern hardware. However, its practical effectiveness depends on low-level software support, including compiler and kernel optimization. In addition, pattern pruning may still perform poorly in layers such as 1×1 convolutions or fully connected layers [42], [44].

VI. Criteria for Selecting Structures

One of the central problems of pruning is determining the importance of structures that compose a model. To achieve this, many works define a criterion to assign an importance score to each structure, allowing a wide range of approaches for pruning methods [32], [33], [39]. One prevalent technique for estimating structure importance considers the magnitude of weights, namely ℓ_p-norm [18], [25]. Despite its intuitiveness, previous works pointed out limitations regarding its fragility to small magnitudes (i.e., noise) and subsequently, low variance of scores [45], [46]. In this context, Lee et al. [46] proposed rescaling the importance of weights by the cumulative magnitude of all surviving parameters. Through this process, the authors employ the ℓ_2-norm more effectively than other related techniques and achieve a better global pruning rate. Overall, since ℓ_p-norm criteria are independent of the training data, the literature categorizes them as data-agnostic.

Apart from the limitations above, data-agnostic criteria fail to capture the relationship with data, motivating studies towards more complex strategies. In this context, data-driven techniques leverage information from representations encoded in the learning phase. For example, Lin et al. [33] assign importance to filters based on the rank of feature maps. An alternative strategy is to estimate the impact of removing a structure on the loss function using its Taylor expansion. For example, Liu et al. [32] approach the problem through the lens of Fisher information, combining the average gradient across samples with memory improvements. This method enables the removal of filters that are not only unimportant but also lead to higher speedups. Shen et al. [15] reformulate the problem of maximizing accuracy under a computational budget as a resource allocation task to solve via the Knapsack paradigm.

While the aforementioned criteria employ hand-crafted heuristics or optimization strategies, studies suggest that random approaches can match the performance of sophisticated methods criteria at high compression rates [47]–[49]. This counterintuitive and unexpected behavior opens room for novel methods that rethink pruning criteria.

Takeaways. The effectiveness of pruning depends heavily on the criterion for estimating importance and consequently on defining structures to remove. Data-agnostic approaches offer simplicity and efficiency but often overlook critical interactions with data. In contrast, data-driven techniques provide more accurate assessments. Interestingly, random criteria challenge the idea that complex heuristics always perform better, especially under high compression.

VII. Parameter Adjustments after Pruning

Adjusting parameters after pruning is a critical step to ensure that pruned networks recover their predictive capacity. Early studies support that pruning reduces the capacity of subnetworks [16], [23], [50], [51]. Fortunately, a few epochs of fine-tuning restore model capacity to a level close to or even exceeding the original network [50].

Previous works introduce different approaches to address parameter adjustment, ranging from conventional fine-tuning techniques [18], [52]–[54] to advanced learning rate schedulers [25], [55] and distillation methods [36], [37], [56]. This line of research is particularly important because pruning often introduces substantial changes to the model architecture, requiring a rebalancing of parameters to allow the model to adapt to its new (pruned) configuration. Standard supervised fine-tuning remains the most widely used and effective approaches for adjusting pruned networks. Overall, it involves continuing the training of the model after pruning, allowing the network parameters to adapt to the new reduced architecture. The seminal work by Han et al. [18] shows that fine-tuning plays a vital role in restoring model capacity under high compression regimes. In this direction, studies confirm that selecting appropriate learning-rate schedulers and regularization techniques makes fine-tuning more effective after pruning [48], [54]. For example, Le et al. [48], demonstrate that dynamic learning-rate schedulers consistently outperform fixed learning rates.

As an alternative to fine-tuning, studies also employ KD methods to restore the performance of pruned models. For example, Chen et al. [36] describe a pruning pipeline where the unpruned model acts as the teacher. It guides the pruned model through an additional loss term that measures the difference between their respective predictions. Interestingly, Muralidharan et al. [16] demonstrate that distillation is superior to conventional fine-tuning when adjusting Large Language Models, establishing it as the preferred approach.

Takeaways. Parameter adjustment is crucial to recover capacity after pruning a model [6]. Among existing approaches, standard supervised fine-tuning is a common choice; however, its efficiency depends on optimal choices of learning rate schedulers and regularization techniques. Alternatively, knowledge distillation methods offer a valuable approach to recovering capacity after pruning. Regardless of the technique, adjusting parameters introduces additional cost since pruned models, despite being more efficient, still require training [6]. Such a cost might be prohibitive to resource-constrained scenarios and, therefore, should be taken into account when evaluating the overall efficiency of a pruning method.

VIII. Pruning Large Vision and Language Models

Given the rapid evolution and adoption of Large Language and Vision Models, most recent advances in pruning research now focus on these architectures. Within this family of models, pruning mechanisms span a spectrum of granularities we covered before, from coarse-grained block removal to fine-grained element-level sparsity [10], [11], [23], [57].

On Large Language Models, structural pruning techniques operate by identifying and selectively removing coupled neuron groups that minimally perturb the loss function value [58]. Then, Low-Rank Adaptation (LoRA) [59] recovers the subsequent performance degradation. Parallel efforts focus on reducing the dimensionality of individual Transformer blocks [57], [60]. For example, Gao et al. [60] remove rows and columns from blocks by projecting feature maps and subsequently re-expanding these maps to preserve dense matrix configurations. Lin et al. [11] advance these principles with a method that decomposes MLP and attention head modules into sub-components. They then approximate each component via Nyström, Column-Row, or SVD matrix factorization. This approach achieves compression ratios of up to 30% with a negligible drop in predictive ability. Conversely, Kim et al. [10] remove layers based on Taylor-series-based loss estimations or perplexity deltas. These criteria serve to identify blocks with minimal contributions to the model's predictive performance.

At the other end of the granularity spectrum, element-level pruning methods focus on inducing high degrees of sparsity. Techniques in this sphere often achieve sparsity levels exceeding 50% [13], [23], [61]. These methods employ sophisticated criteria for ranking individual weights, such as (i) activation-weighted magnitudes [23]; (ii) Hessian-guided sparse regression [24]; and (iii) weight and activation information [61].

On image-language and video-language understanding benchmarks, Alvar et al. [12] address token redundancy by solving a Max-Min diversity problem. Their method removes up to 45% of computation with negligible loss in predictive performance. In a similar vein, Liang et al. [13] compress multimodal models using only a few examples by searching for prunable configurations that maintain alignment between visual and textual representations. Unlike most strategies, Sun et al. [14] assign importance to retaining a given total number of structures such as projection matrices, embeddings, and blocks. They adopt a Mixed-Integer Nonlinear Programming formulation to remove components globally under latency constraints. On 3D object detection, their method achieves up to 1.8× speed-up while increasing detection quality.

As we mentioned in Section VII, parameter adjustment in pruning may add a notable cost during model compression. To mitigate this issue, Lou et al. [6] introduce an error propagation scheme to compensate for the removal of modules with minimal fine-tuning steps. Specifically, their method uses the subsequent unpruned layer to adjust the weights of a given pruned layer. Overall, their method exhibits positive results on instance segmentation using the SAM model family, even at compression rates above 50%. In common-sense language tasks, the method demonstrates the same behavior.

Takeaways. The success of modern pruning on large models hinges on well-designed criteria for avoiding model collapse and generalization on downstream tasks. Unlike the vast literature on pruning in image classification [5], [19], studies suggest that iteratively pruning and adjusting pruned models to achieve a given compression ratio (iterative pruning) performs worse than one-shot pruning [13], [24], [61]. Due to the high computational cost, most works avoid standard fine-tuning and instead focus on PEFT mechanisms (e.g., LoRA [59]) for pruning large models, as these techniques recover model capacity while imposing minimal additional hardware overhead.

IX. Conclusions and Future Trends

Large Vision, Language and Multimodal Models introduce a paradigm shift in AI capabilities [1]–[3]. These models often include hundreds of billions of parameters, imposing substantial computational overheads. Deploying this family of models demands substantial energy, leading to high carbon emissions and operational costs [3], [4]. Pruning offers a compelling strategy to reduce these overheads by lowering computational requirements and minimizing related costs like energy consumption and financial expenses. By eliminating structures, pruning techniques reduce computing, memory usage, and inference latency- Unlike methods that require architectural redesigns, pruning integrates seamlessly into off-the-shelf models and existing pipelines.

Despite progress toward more efficient models, pruning research still faces many open questions. First, some forms of pruning rely on careful software support to translate theoretical gains into meaningful practical benefits. Second, it is still unclear whether well-designed criteria and optimization strategies actually outperform simple random heuristics. Third, pruning modern large models directly in resource-constrained environments (i.e., consumer-grade GPUs) remains challenging due to the parameter adjustment phase.

Beyond these points, we believe more effort should focus on simultaneously removing multiple structures—weights, filters, and blocks—leveraging the advantages of each pruning granularity while addressing their individual limitations. Finally, current literature lacks studies on the effectiveness of pruning in multi-modal models that jointly process vision and language. Therefore, we believe this is a key line of research for enabling efficient general-purpose AI models.

Acknowledgments

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. Aline Paes would like to thank CNPq (National Council for Scientific and Technological Development), grant 307088/2023-5, FAPERJ - *Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro*, processes SEI-260003/002930/2024, SEI-260003/000614/2023.

References

[1] N. Maslej and et al., "Artificial intelligence index report," 2024.

[2] Y. Bengio and et al., "International AI safety report," 2025.

[3] J. Morrison, , and et al., "Holistically evaluating the environmental impact of creating language models," in *ICLR*, 2025.

[4] A. Faiz and et al., "Llmcarbon: Modeling the end-to-end carbon footprint of large language models," *ICLR*, 2024.

[5] H. Cheng and et al., "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE TPAMI*, 2024.

[6] X. Luo and et al., "Icp: Immediate compensation pruning for mid-to-high sparsity," in *CVPR*, 2025.

[7] D. Mittal and et al., "Recovering from random pruning: On the plasticity of deep convolutional neural networks," in *WACV*, 2018.

[8] Y. LeCun and et al., "Optimal brain damage," in *NeurIPS*, 1989.

[9] B. Hassibi and et al., "Second order derivatives for network pruning: Optimal brain surgeon," in *NeurIPS*, 1992.

[10] B.-K. Kim and et al., "Shortened llama: A simple depth pruning for large language models," in *ICLR*, 2024.

[11] C. Lin and et al., "Modegpt: Modular decomposition for large language model compression," in *ICLR*, 2025.

[12] S. R. Alvar and et al., "Divprune: Diversity-based visual token pruning for large multimodal models," in *CVPR*, 2025.

[13] Y. Liang and et al., "Efficientllava:generalizable auto-pruning for large vision-language models," in *CVPR*, 2025.

[14] X. Sun and et al., "MDP: multidimensional vision model pruning with latency constraint," in *CVPR*, 2025.

[15] M. Shen and et al., "Structural pruning via latency-saliency knapsack," in *NeurIPS*, 2022.

[16] S. Muralidharan and et al., "Compact language models via pruning and knowledge distillation," in *NeurIPS*, 2024.

[17] A. Ganjdanesh and et al., "Jointly training and pruning cnns via learnable agent guidance and alignment," in *CVPR*, 2024.

[18] S. Han and et al., "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015.

[19] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE TPAMI*, 2023.

[20] E. Elsen and et al., "Fast sparse convnets," in *CVPR*, 2020.

[21] W. Luo and et al., "Benchmarking and dissecting the nvidia hopper GPU architecture," in *IPDPS*, 2024.

[22] B. Hassibi and et al., "Optimal brain surgeon and general network pruning," in *ICNN*, 1993.

[23] M. Sun and et al., "A simple and effective pruning approach for large language models," in *ICLR*, 2024.

[24] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," in *ICML*, 2023.

[25] J. Frankle and et al., "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2019.

[26] M. Shen and et al., "When to prune? A policy towards early structural pruning," in *CVPR*, 2022.

[27] H. Wang and et al., "Recent advances on neural network pruning at initialization," in *IJCAI*, 2022.

[28] A. Zhou and et al., "Learning N: M fine-grained structured sparse neural networks from scratch," in *ICLR*, 2021.

[29] W. Niu and et al., "Patdnn: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning," in *ASPLOS*, 2020.

[30] H. Li and et al., "Pruning filters for efficient convnets," in *ICLR*, 2017.

[31] Y. He and et al., "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *CVPR*, 2020.

[32] L. Liu and et al., "Group fisher pruning for practical network compression," in *ICML*, 2021.

[33] M. Lin and et al., "Hrank: Filter pruning using high-rank feature map," in *CVPR*, 2020.

[34] A. Gupta and et al., "Torque based structured pruning for deep neural network," in *WACV*, 2024.

[35] A. Veit, M. J. Wilber, and S. J. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *NeurIPS*, 2016.

[36] S. Chen and Q. Zhao, "Shallowing deep networks: Layer-wise pruning based on feature representations," *IEEE TPAMI*, 2019.

[37] Y. Zhou, G. G. Yen, and Z. Yi, "Evolutionary shallowing deep neural networks at block levels," *IEEE TNNLS*, 2022.

[38] C. Zhang, S. Bengio, and Y. Singer, "Are all layers created equal?" *Journal of Machine Learning Research*, 2022.

[39] I. Pons, B. Yamamoto, A. H. R. Costa, and A. Jordao, "Effective layer pruning through similarity metric perspective," in *ICPR*, 2024.

[40] A. B. D. et al., "Layer folding: Neural network depth reduction using activation linearization," in *BMVC*, 2022.

[41] Y. F. et al., "Depthshrinker: A new compression paradigm towards boosting real-hardware efficiency of compact neural networks," in *ICML*, 2022.

[42] Z. Li and et al., "NPAS: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration," in *CVPR*, 2021.

[43] X. Ma and et al., "PCONV: the missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices," in *AAAI*, 2020.

[44] Z. Zhan and et al., "Achieving on-mobile real-time super-resolution with neural architecture and pruning search," in *ICCV*, 2021.

[45] Y. He and et al., "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *CVPR*, 2019.

[46] J. Lee and et al., "Layer-adaptive sparsity for the magnitude-based pruning," in *ICLR*, 2021.

[47] P. de Jorge and et al., "Progressive skeletonization: Trimming more fat from a network at initialization," in *ICLR*, 2021.

[48] D. H. Le and et al., "Network pruning that matters: A case study on retraining variants," in *ICLR*, 2021.

[49] Y. Li and et al., "Revisiting random channel pruning for neural network compression," in *CVPR*, 2022.

[50] G. Mason-Williams and et al., "What makes a good prune? maximal unstructured pruning for maximal cosine similarity," in *ICLR*, 2024.

[51] G. H. do Nascimento and et al., "Pruning everything, everywhere, all at once," in *IJCNN*, 2025.

[52] S. Han and et al., "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[53] S. Yamaguchi and et al., "Adaptive random feature regularization on fine-tuning deep neural networks," in *CVPR*, 2024.

[54] D. Li and et al., "Improved regularization and robustness for fine-tuning in neural networks," in *NeurIPS*, 2021.

[55] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *ICLR*, 2020.

[56] D. Chen and et al., "EPSD: early pruning with self-distillation for efficient model compression," in *AAAI*, 2024.

[57] S. Ashkboos and et al., "SliceGPT: Compress large language models by deleting rows and columns," in *ICLR*, 2024.

[58] X. Ma and et al., "Llm-pruner: On the structural pruning of large language models," in *NeurIPS*, 2023.

[59] E. J. Hu and et al., "Lora: Low-rank adaptation of large language models," in *ICLR*, 2022.

[60] S. Gao and et al., "DISP-LLM: Dimension-independent structural pruning for large language models," in *NeurIPS*, 2024.

[61] Y. Z. et al., "Plug-and-play: An efficient post-training pruning method for large language models," in *ICLR*, 2024.

[62] E. Strubell and et al., "Energy and policy considerations for deep learning in NLP," in *ACL*, 2019.