Pruning Everything, Everywhere, All at Once

Gustavo Henrique do Nascimento, Ian Pons, Anna Helena Reali Costa and Artur Jordao
Escola Politécnica, Universidade de São Paulo, Brazil

Abstract—**Deep learning stands as the modern paradigm for solving cognitive tasks. However, as the problem complexity increases, models grow deeper and computationally prohibitive, hindering advancements in real-world and resource-constrained applications. Extensive studies reveal that pruning structures in these models efficiently reduces model complexity and improves computational efficiency. Successful strategies in this sphere include removing neurons (i.e., filters, heads) or layers, but not both together. Therefore, simultaneously pruning different structures remains an open problem. To fill this gap and leverage the benefits of eliminating neurons and layers at once, we propose a new method capable of pruning different structures within a model as follows. Given two candidate subnetworks (pruned models), one from layer pruning and the other from neuron pruning, our method decides which to choose by selecting the one with the highest representation similarity to its parent (the network that generates the subnetworks) using the Centered Kernel Alignment (CKA) metric. Iteratively repeating this process provides highly sparse models that preserve the original predictive ability. Throughout extensive experiments on standard architectures and benchmarks, we confirm the effectiveness of our approach and show that it outperforms state-of-the-art layer and filter pruning techniques. At high levels of Floating Point Operations (FLOPs) reduction, most state-of-the-art methods degrade accuracy, whereas our approach either improves it or experiences only a minimal drop. Notably, on the popular ResNet56 and ResNet110, we achieve a milestone of 86.37% and 95.82% FLOPs reduction. Besides, our pruned models obtain robustness to adversarial and out-of-distribution samples and take an important step towards GreenAI, reducing carbon emissions by up to 83.31%. Overall, we believe our work opens a new chapter in pruning. Code is available at: https://github.com/NascimentoG/PruningEverything.**

I. Introduction

Deep learning drives unprecedented progress in various cognitive tasks and serves as the powerhouse for learning patterns from data [1], [2]. However, this performance incurs a significant computational cost, and consequently environmental impacts [3]. The recent LLama 3 model family comprehends a concrete example, where training requires up to 16K H100 GPUs globally distributed. Notably, the largest model (405B) requires 16 GPUs with 16-bit precision for inference [4].

Efforts towards compression of deep models confirm the effectiveness of pruning methods in reducing computational demands without degrading the predictive ability of models [5]. Most pruning methods remove small structures composing a model such as weights or filters [5], [6]. The first, a.k.a unstructured pruning, obtains higher compression rates but requires specialized hardware for sparse computing. The latter, namely structured pruning, offers hardware-agnostic benefits and competitive sparsity rates. In contrast to these popular forms of pruning, a parallel line of study suggests that removing layers achieves both superior predictive preservation and better enhances depth-related performance metrics such as latency [7], [8]. However, layer pruning is not without its drawbacks. For example, due to technical details, the number of layers available for removal is notably smaller than filters; thus, limiting the maximum compression level this family of pruning can obtain (roughly 75.5% of Floating Point Operations – FLOPs – on the popular CIFAR10 + ResNet56 setting). Figure 1 (**Left**) illustrates this behavior, where a typical layer-pruning method consistently removes layers until none remain among those eligible for removing. This means they take the path to the right until reaching a leaf (see symbol * in Figure 1 (**Left**)). As a concrete example, Pons et al. [7] eliminate layers until no more layers are available for pruning. After this point, to achieve even higher compression rates, their method starts removing neurons (i.e., filters), following the path to the left from the leaf (indicated by *) in Figure 1 (**Left**). Technically speaking, existing filter-pruning approaches share similar limitations. Considering that the literature focuses on eliminating individual structures [5], [6], filters or layers but not both, a naive strategy for obtaining high-level compression is to eliminate different structures throughout a pruning process.

From the previous discussion, a possible approach to overcome the limitations of existing structured pruning involves alternating between structures. Thus, a natural question arises: *how do we decide which structure to remove during pruning?* Figure 1 (**Left**) allows restructuring the question as follows: *given a network (the parent), should we take the right path (removing layers) or the left path (removing filters) as pruning progresses?*

To accomplish this, we propose systematically deciding between eliminating layers or filters as pruning progresses. An overview of our method is the following. From a network, we generate two candidate subnetworks: one through layer and filter pruning, respectively. Afterwards, we use Centred Kernel Alignment (CKA) [9] to compare the internal representation of the candidates with the network that generated them (namely, parent). Then, we take the candidate with higher similarity. From the perspective of Figure 1 (**Left**), we use CKA to guide our choice between going to the left (eliminate filters) or right (eliminate layers) path for a given network. The rationality behind this process involves evaluating how well a candidate subnetwork preserves the representation of its parent: greater similarity indicates a proper candidate.

The process above comprehends one iteration of our method. Following previous works [7], [10]–[12], we perform this process iteratively and, for each iteration, our method

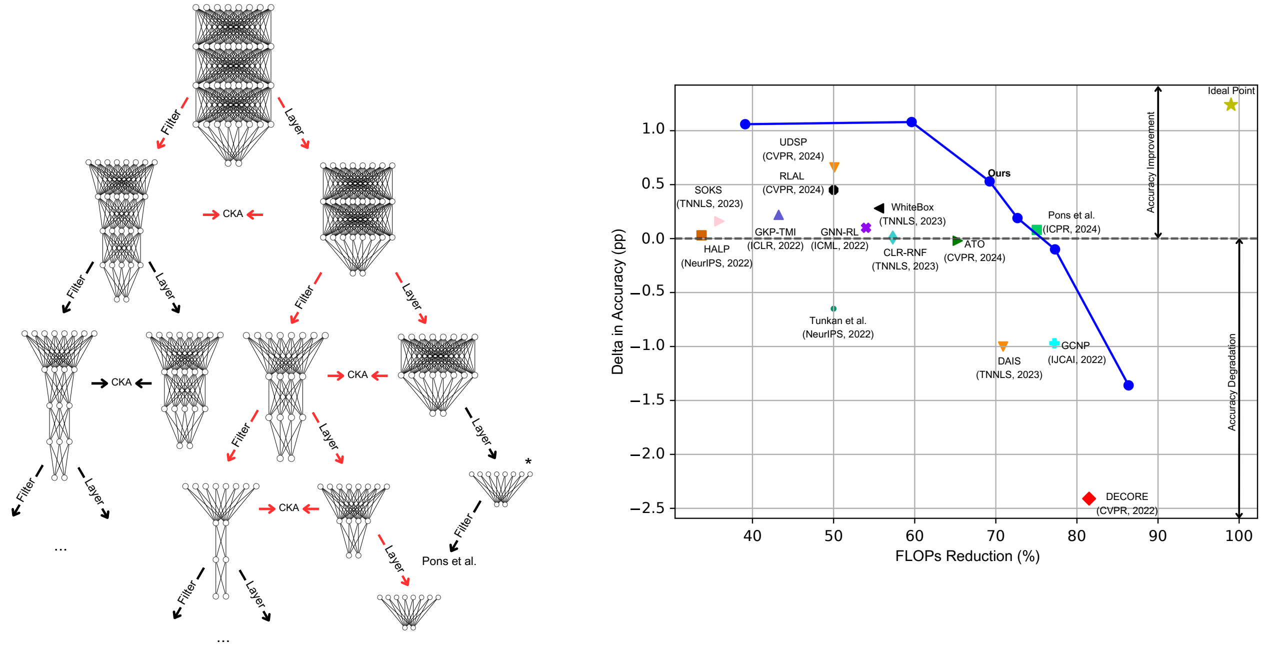

Fig. 1. **Left.** A binary decision tree of potential pruning outcomes and the paths (red arrows) to follow using our approach. The process starts with an unpruned deep network (parent), for which we generate two subnetworks (pruned): one from layer pruning and the other from filter pruning. Then, with a similarity metric (i.e., Centred Kernel Alignment – CKA), we evaluate the resulting subnetworks to determine and select the one that better preserves the representation of its parent. Our method iteratively repeats this strategy until no more structures (filters or layers) remain for removal or until reaching a user-defined number of iterations. To accomplish this, the subnetwork with the highest similarity to its parent (i.e., the selected subnetwork candidate) becomes the parent for the next iteration (the next depth level in the tree). The example illustrates our method over four pruning iterations, where it systematically eliminated structures in the following sequence: layers, filters, layers, and layers. **Right.** Comparison with state-of-the-art techniques on the standard ResNet56 + CIFAR-10 setting. Our method significantly outperforms existing layer and filter pruning techniques by a large margin. Specifically, compared to other pruning approaches, our method removes over 80% of FLOPs (Floating Point Operations) with negligible delta in accuracy. In contrast, other methods degrade accuracy when operating at such high levels of FLOPs reduction. In summary, our method achieves better trade-offs in accuracy and computational efficiency (measured by FLOPs). These results confirm how our approach enables better pruning by making informed choices between layers and filters, surpassing most state-of-the-art techniques. (For illustration purposes only, we intentionally misuse notation and indicate the ideal point on the top right.)

chooses the candidate greedily. This means it selects the proper candidate at each pruning iteration (one level in Figure 1 (**Left**)) and ensures computational efficiency with $O(log(n))$, where n is the set of all possible candidate subnetworks.

Research Statement and Contributions. Overall, our work has the following research statement: *Given two candidate subnetworks (pruned models), one from layer pruning and the other from filter pruning, we can effectively decide which to choose by selecting the one with the highest representation similarity to its parent. Iteratively and greedily repeating this process provides highly sparse models that preserve the predictive ability of the overparameterized original model (unpruned).* We confirm this statement across a broad range of experimental settings, datasets, and architectures.

Among our contributions, we highlight the following. We introduce a new form of pruning that eliminates both filter and layer from deep models and leverages the best of existing techniques. From a practical perspective, our method contributes to achieving a new milestone of computational gains. Our method is not confined to removing only one type of structure and integrates the benefits of both filter and layer methods. From a theoretical perspective, we demonstrate that preserving representation similarity between the parent model and a candidate subnetwork (from layer and filter pruning) indicates an effective strategy for systematically deciding which structure to eliminate. This enables maintaining predictive ability during pruning, even at significantly high compression rates.

Figure 1 (**Right**) supports the previous statements by introducing our results alongside other state-of-the-art methods. Specifically, we achieve a 72.67% reduction in FLOPs without degrading accuracy and reach a milestone of 86.37% with a negligible drop. Extensive experiments across a broad range of benchmarks and architectures confirm our method outperforms state-of-the-art pruning techniques. Specifically, we surpass existing methods by achieving 61.99% and 71.31% FLOPs reduction on ResNet32 and ResNet44 while increasing accuracy. On deeper architectures, we obtain a milestone of 86.37% and 95.82% FLOPs reduction on ResNet56/110. In terms of GreenAI [3], [13], these results comprehend an important step toward reducing carbon emissions linked to energy consumption during model deployment. Our pruned models also reduce financial costs associated with training and fine-tuning by up to 83%.

With the popularization of (large) foundation models [1], [2] and the increasing need for efficient models, we believe our results open a new chapter for achieving higher levels of compression by breaking down the barriers of existing strategies that rely solely on one structure.

II. Related Work

Recent literature categorizes this family of methods into two approaches [5], [6]: structured and unstructured. Structured pruning removes structural components of the neural network, such as neurons (filters, attention heads) and layers. Overall, these methods assign a score to each structure based on specific criterion and remove structures accordingly. In contrast, unstructured pruning focuses on individual weights. Although this form of pruning achieves higher sparsity levels, deploying it in real-world scenarios demands specific hardware support, whereas structured pruning avoids such constraints. Additionally, most pruning methods optimize for specific structures, such as filters or layers, being unable to leverage the possibility of removing more than one type of structure (i.e., orthogonality) [5], [6]. As a concrete example, Gao et al. [14] proposed a bi-level optimization to integrate benefits of both efficiency and storage. To accomplish this, their method combines dynamic (input-dependent, adapting during inference) and static (unchanged during inference) filter pruning. From the lens of layer pruning, Ganjdanesh et al. [12] addressed the problem with reinforcement learning (RL), pruning iteratively while updating both RL and model parameters. Wu et al. [15] proposed a method for pruning while working alongside a controller network to avoid falling into local optima. Kim et al. [8] compressed models by using a layer merge method that removes layers and activation functions. Yang et al. [16] introduced a method that collapses later layers into earlier ones, reducing model size while preserving its structure. In a similar direction, Gromov et al. [17] focused on analyzing where knowledge is stored: in deeper or shallower layers.

Despite positive results, the aforementioned works focus on pruning only one type of structure. Therefore, removing both structures simultaneously remains an open problem. In contrast, our method fills this gap. It is worth mentioning that the work by Pons et al. [7] constitutes a preliminary effort in this direction. Specifically, the authors introduced a layer-pruning technique. Unfortunately, the number of layers constrains the achievable computational benefit. To mitigate this problem, their method leverages the orthogonal nature of filter and layer pruning and begins removing filters only after eliminating all possible layers. However, the authors observed that this approach fails to preserve generalization, highlighting the complexity of synergizing filter and layer pruning. This supports the core idea behind our method as it carefully and systematically chooses among the type of structure.

Closely related to ours, Muralidharan et al. [10] discover subnetworks by alternating between width (i.e., filters and attention heads) and depth (i.e., layers) pruning. Unfortunately, their process relies on an empirical trial-and-error mechanism. The challenge with this strategy lies in understanding the process behind deriving these rules and determining their applicability to all models. In contrast, our method follows a systematic and empirical-agnostic approach by employing similarity metrics to select subnetworks from filter or layer pruning automatically. Therefore, we believe our work opens a new chapter in pruning by exploring structural possibilities as decisions, thereby overcoming the inherent limitations of structure-dependent pruning.

III. Preliminaries and Proposed Method

Problem Statement. According to recent literature [5], [6], [18], layer and filter strategies offer diverse and complementary advantages in reducing the computational cost of deep models. For example, filter pruning enables a high compression rate while layer pruning promotes notable inference improvements. Therefore, to leverage the best of both worlds, we define the following problem statement: *how to identify the most promising structure to remove across an iterative pruning process?* Solving this problem enables obtaining shallower (layer pruning) and narrower (filter pruning) models after repeating the process multiple times.

The above formulation naturally suggests a naive solution: if we aim to remove layers or filters during a pruning process, why not just choose randomly which one to eliminate? Previous studies endorse this naive solution [19], confirming the potential of random pruning. We demonstrate that systematically and carefully selecting among layers or filters (as explained below) leads to better results for this problem.

Definitions. Let $\mathcal{F}$ denote a neural network trained on a dataset $\mathbf{X} = \{x_i\}_{i=1}^{D}$ with corresponding labels $\mathbf{Y} = \{y_i\}_{i=1}^{D}$, where D is the size of the dataset. Consider a pruning criterion c that assigns a score (i.e., importance) to each structure (filter or layer) composing the network. Based on these scores, a pruning algorithm eliminates the less important structures.

Applying c to a neural network to prune filters results in a subnetwork $\mathcal{F}_f^{'}$. Similarly, using it to prune layers produces a subnetwork $\mathcal{F}_l^{'}$. As we shall see, the core of our method relies on comparing these pruned networks with their parent. To accomplish this, consider a similarity metric that takes the internal representations (feature maps) of two neural networks as input and provides a measure of their similarity. In this work, we employ the Centered Kernel Alignment (CKA) similarity metric [9], defined as the normalized Hilbert-Schmidt Independence Criterion (HSIC) [20]. Formally, let k and l be two kernels, the empirical estimator of HSIC with m examples for the internal representation extraction is:

$$HSIC(K, L) = \frac{1}{(m-1)^2} tr(KHLH), \tag{1}$$

where $K_{ij} = k(x_i, x_j)$, $L_{ij} = l(y_i, y_j)$ and $H_{ij} = \delta_{ij} - m^{-1}$. Here, $\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$ otherwise.

Let R and $R_{\mathcal{F}'}$ be the internal representation from the parent model $\mathcal{F}$ and pruned model $\mathcal{F}^{'}$, respectively. Following Kornblith et al. [9], we compute the CKA in terms of

$$CKA(R, R_{\mathcal{F}'}) = \frac{HSIC(R, R_{\mathcal{F}'})}{\sqrt{HSIC(R, R) \times HSIC(R_{\mathcal{F}'}, R_{\mathcal{F}'})}}. \tag{2}$$

The output from $CKA(R, R_{\mathcal{F}'}) \in [0, 1]$, where higher values indicate greater similarity between the representations.

For simplicity, we abuse notation and apply CKA using two networks directly as inputs – the parent model and its pruned version. In practice, however, CKA compares two matrices that characterize the internal representation of the models.

Proposed Method. Building upon the previous definition, our method operates as follows. First, it produces a layer $\mathcal{F}_l^{'}$ and filter $\mathcal{F}_f^{'}$ pruned version (subnetworks) from a network $\mathcal{F}$. At this step, we follow previous work and generate subnetworks with similar capacity [21], ensuring a consistent comparison among them. Additionally, to ensure a fair comparison, we generate the subnetworks using the same pruning criterion c. An important point regarding c arises here: an effective criterion for removing filters may be inadequate when applied directly to remove layers [7], [21]. To deal with this issue, we apply the KL-divergence criterion by Luo et al. [22]. This criterion measures Kullback-Leibler divergence between the softmax distribution of the unpruned network and the network without a potential component. Therefore, it applies to any structure (weight, filter or layer) and is unaffected by common pitfalls of traditional filter criteria (e.g., ℓ_1-norm) when applied to layers [7]. Interestingly, in Appendix VI-A, we evaluate our method using the popular ℓ_1-norm criterion [5], [6] and confirm it is effective regardless of the criterion.

After the above steps, our method considers $\mathcal{F}_l^{'}$ and $\mathcal{F}_f^{'}$ as pruning candidates and must select one of them. In other words, these candidates form a branch in a binary decision tree (Figure 1 (**Left**)), and the method must choose a path. To accomplish this, the central idea of our method is to select the one with the highest similarity w.r.t $\mathcal{F}$ (with the network that produced them – parent); thereby preserving predictive capability of the original neural network $\mathcal{F}$ at the root of the tree (Figure 1 (top)). To guide this decision, we apply the CKA similarity metric (Equation 2) and formulate the expression:

$$\mathcal{F} \leftarrow \begin{cases} \mathcal{F}_l^{'} & \text{if } CKA(\mathcal{F}, \mathcal{F}_l^{'}) \geq CKA(\mathcal{F}, \mathcal{F}_f^{'}) \\ \mathcal{F}_f^{'} & \text{otherwise.} \end{cases} \tag{3}$$

Due to the iterative nature of our method (details below), in Equation 3, we slightly abuse notation and assign $\mathcal{F}$ to its pruned version for easy of exposition. Finally, we apply a standard (i.e., supervised) fine-tuning, as this step helps recover predictive ability after pruning [10], [18], [23]. Note that when the similarity of both subnetworks is equal, our method prefers to remove the layer, as this form of pruning promotes additional performance gains [7], [18]. In this sense, Equation 3 allows us to easily incorporate inference-aware thought layer pruning by simply penalizing the left part of with a constant ϵ (i.e., $CKA(\mathcal{F}, \mathcal{F}_f^{'}) + \epsilon$).

Algorithm 1 summarizes the steps composing our method. From Algorithm 1, we highlight two key observations. (i) Line 4 indicates a branch (i.e., decision) in Figure 1 (**Left**). (ii) The input to the next $(k+1)$ pruning iteration is the model pruned in the previous (k) iteration, see lines 5 and 7. Importantly, this is what ensures the algorithm produces shallower and narrower models after multiple iterations (K).

Regarding the time-complexity, since for each iteration (k) we reduce the problem in terms of $\frac{n}{2^K}$, where n is the number of nodes in the tree, our method has $O(log(n))$ time complexity. In this direction, we iterate using a sufficiently large K; thus, obtaining notably shallow and narrow models.

Algorithm 1 Proposed Method

```
    Input: Trained Neural Network $\mathcal{F}$, Pruning Criterion $c$, Number of Iterations $K$, Training samples $\mathbf{X}$ and the respective labels $\mathbf{Y}$
    Output: Pruned version of $\mathcal{F}$
 1: for $k \leftarrow 1$ to $K$ do
 2:     $\mathcal{F}_l^{'} \leftarrow \mathcal{F} \setminus c(\mathcal{F})$ ▷ Remove unimportant layers based on $c$
 3:     $\mathcal{F}_f^{'} \leftarrow \mathcal{F} \setminus c(\mathcal{F})$ ▷ Remove unimportant filters based on $c$
 4:     if $CKA(\mathcal{F}, \mathcal{F}_l^{'}) \geq CKA(\mathcal{F}, \mathcal{F}_f^{'})$ then
 5:         $\mathcal{F} \leftarrow \mathcal{F}_l^{'}$ ▷ $\mathcal{F}$ becomes its layer pruned version
 6:     else
 7:         $\mathcal{F} \leftarrow \mathcal{F}_f^{'}$ ▷ $\mathcal{F}$ becomes its filter pruned version
 8:     end if
 9:     Update $\mathcal{F}$ via standard fine-tuning on $\mathbf{X}$ and $\mathbf{Y}$
10: end for
```

IV. Experiments

Experimental Setup. We conduct the experiments on CIFAR-10 and ImageNet using different ResNet architectures [24]. These settings are a standard choice to assess the effectiveness of pruning [6], [7], [12], [14], [15]. Following previous works [10], [23], [25] before comparing the two pruned subnetworks in Algorithm 1 (line 4), we also perform a 10-epoch fine-tuning. We omit this step from Algorithm 1 to maintain the simplicity and clarity of our approach.

Overall, we keep the experimental setup as simple as possible to highlight the potential and advantages of our method compared to more elaborate setups on pruning such as knowledge distillation [10], [26], [27]. We also conduct experiments on Transformers and detail our findings in Appendix VI-C.

To summarize the results, we adopt common approaches and present the delta in accuracy in percentage points (pp) [6]. As shown in Figure 1 (**Right**), a negative delta indicates a degradation in accuracy, while a positive delta reveals an improvement. Finally, for each division of FLOP reduction (%), we highlight the best results in bold in terms of delta in accuracy and FLOP reduction, separately.

Comparison with State-of-The-Art. We start our analysis by comparing our method with modern and top-performance pruning techniques. Table I summarizes the results on the standard FLOPs reduction and delta in accuracy compromises.

According to Table I, our approach surpasses most existing techniques, regardless of whether it is pruned by layers or filters. On ResNet56, our best result (without degrading generalization) reduces 72.67% of FLOPs while improving accuracy in 0.19 pp. In terms of maintaining positive accuracy while achieving high compression, our method only underperform Pons et al. [7]. Notably, their method is confined to not exceeding this value, as there are no more layers to remove (see the symbol * in Figure 1 (**Left**)). In particular, all methods suffer from this constraint since they are one structure pruning. It turns out that while state-of-the-art techniques opt to remove only layers (path to the right) or only filters (path to the left), they overlook the middle ground. Focusing on only one type of

TABLE I
COMPARISON WITH STATE-OF-THE-ART METHODS ON CIFAR-10.

	Method	Δ Acc.	FLOPs (%)
ResNet56	**Ours**	**(+) 0.91**	53.70
	ATO [15] (CVPR, 2024)	(+) 0.24	55.00
	WhiteBox [28] (TNNLS, 2023)	(+) 0.28	**55.60**
	CLR-RNF [29] (TNNLS, 2023)	(+) 0.01	57.30
	Pons et al. [7] (ICPR, 2024)	(+) 0.78	60.04
	Ours	**(+) 0.96**	**62.79**
	ATO [15] (CVPR, 2024)	(-) 0.02	65.30
	Ours	**(+) 0.53**	**69.24**
	DAIS [30] (TNNLS, 2023)	(-) 1.00	70.90
	Ours	**(+) 0.19**	72.67
	Pons et al. [7] (ICPR, 2024)	(+) 0.08	75.05
	GCNP [31] (IJCAI, 2022)	(-) 0.97	77.22
	Ours	(-) 0.49	**79.88**
	DECORE [32] (CVPR, 2022)	(-) 2.41	81.50
	Ours	**(-) 1.36**	**86.37**
ResNet110	DECORE [32] (CVPR, 2022)	(+) 0.38	35.43
	GKP-TMI [33] (ICLR, 2022)	(+) 0.64	43.31
	Pons et al. [7] (ICPR, 2024)	(+) 1.37	44.73
	Ours	**(+) 1.47**	**48.29**
	Pons et al. [7] (ICPR, 2024)	**(+) 0.89**	65.23
	CRL-RNF [29] (TNNLS, 2023)	(+) 0.14	66.00
	WhiteBox [28] (TNNLS, 2023)	(+) 0.62	66.00
	Pons et al. [7] (ICPR, 2024)	(+) 0.80	67.10
	Ours	(+) 0.86	**69.90**
	Pons et al. [7] (ICPR, 2024)	(+) 0.59	70.83
	Ours	**(+) 0.60**	**73.16**
	Ours	**(+) 0.49**	75.55
	DECORE [32] (CVPR, 2022)	(-) 0.79	76.92
	Pons et al. [7] (ICPR, 2024)	(+) 0.23	**76.42**
	Pons et al. [7] (ICPR, 2024)	**(-) 0.41**	87.61
	Ours	(-) 2.91	**95.82**

structure ignores the fact that there is much more to explore by systematically alternating between them. The results in Table I confirm the superiority of making informed pruning decisions. When considering only FLOP reductions, our approach outperforms all other works, achieving a reduction of 86.37%.

On ResNet110, the previous behavior repeats itself, and considering the level of compression, our approach achieves a FLOP reduction of 95.82% while exhibiting a negligible delta in accuracy. To the best of our knowledge, these values represent the highest reduction achieved solely through pruning, and recent surveys confirm this [5], [6]. Finally, Table II introduces our results on ImageNet using ResNet50, where we achieve performance on par with or surpassing the state-of-the-art across different ranges of FLOP reduction.

The key factor behind the aforementioned breakthroughs is the careful selection of which structures to remove during the pruning process. Manson et al. [23] argue that knowing which parameters to remove is more important than quantity, reinforcing the argument behind our method, although it explores a structural perspective. Besides, the concept of pruning seeks to maintain generalization while enhancing compression, resulting in a shallower and/or narrower model. By using CKA as a similarity metric, we focus directly on the internal representation of the neural network. When we adopt it as a selection criterion, we retain the best subnetwork that represents its parent in terms of internal representation, allowing continued pruning while preserving accuracy.

TABLE II
COMPARISON WITH STATE-OF-THE-ART PRUNING METHODS ON IMAGENET. Δ ACC. CONSIDERS TOP1 ACCURACY.

	Method	Δ Acc.	FLOPs (%)
ResNet50	DECORE [32] (CVPR, 2022)	(+) 0.16	13.45
	Ours	**(+) 2.09**	**16.98**
	GKP-TMI [33] (ICLR, 2022)	(-) 0.19	22.50
	Pons et al. [7] (ICPR, 2024)	(+) 1.11	22.64
	Ours	**(+) 1.61**	**22.64**
	Pons et al. [7] (ICPR, 2024)	(+) 0.74	28.30
	Ours	**(+) 1.07**	**28.30**
	CLR-RNF [29] (TNNLS, 2023)	(-) 1.16	40.39
	DECORE [32] (CVPR, 2022)	(-) 1.57	42.30
	Ours	(-) 0.98	45.28
	Pons et al. [7] (ICPR, 2024)	(-) 0.90	45.28
	WhiteBox [28] (TNNLS, 2023)	**(-) 0.83**	**45.60**

Following by our findings, the previous results highlight the potential of employing our systematic approach that alternates between layers and filters, rather than solely pruning one type of structure. Moreover, we confirm our research statement: the similarity between the subnetwork and its parent allows to obtain highly sparse models that preserve the predictive ability of the overparameterized original (unpruned) model.

We now turn our attention to analyzing the effectiveness and robustness of the proposed method in different settings.

Effectiveness in Shallow Architectures. One of the keys to successful pruning lies in the overparameterized nature of networks, particularly in deep models. The previous experiments confirm the effectiveness of our method on this family of models. In this experiment, we highlight how the proposed method operates well to shallow architectures. For this purpose, we use the shallow versions of ResNet: ResNet32/44.

By applying our method on ResNet32 and ResNet44, we observe the same behavior as in their deeper versions (ResNet56/110). More concretely, we reduce 61.99% and 71.31% of the FLOPs on ResNet32 and ResNet44, respectively, without compromising accuracy. Interestingly, to obtain more than 95% FLOP reduction on ResNet32, our method exhibited a notable drop in accuracy of 7.63 pp. We highlight that this is the only instance in which we observe a severe drop through our experiments, yet no previous works report such an achievement on this architecture.

Similarly to the deeper models, we also compare these results with state-of-the-art methods for the smaller architectures. Our approach accomplishes notable results, outperforming existing methods, and pushes the boundaries of compression, reducing 87.06% and 83.19% of the FLOPs on ResNet32/44, with only a small drop in accuracy. We provide detailed results in Appendix VI-B. Overall, our method performs well on both deep and shallow networks and is compatible with modern architectures, as we show in Appendix VI-C.

Is CKA Better than Random Choice? As we mentioned in our problem definition, a naive solution for selecting between eliminating filters and layers during an iterative pruning process is simply to choose randomly (i.e., a random selection). In this experiment, we demonstrate that random selection is inefficient and leads to poorly pruned models. For

TABLE III
COMPARISON WITH RANDOM SELECTION BASELINE (RANDOM WALK) ON CIFAR-10. ALTHOUGH RANDOM WALK HIGHLIGHTS POSITIVE ACCURACY, IT STOPS PRUNING EARLIER.

	Method	Δ Acc.	FLOPs (%)
ResNet32	Random Walk	+0.12	55.83
	Ours	+0.09	**61.99**
ResNet44	Random Walk	+0.21	69.74
	Ours	+0.66	**71.31**
ResNet56	Random Walk	+0.19	69.64
	Ours	+0.19	**72.67**

this purpose, we prune the architectures ResNet32/44/56 by randomly selecting whether to remove filters or layers (namely, Random Walk). Importantly, we compare our method with Random Walk, varying only the selection mechanism, while all other settings remain identical. From these settings, we force pruning until the pruned model exhibits no degradation compared to the unpruned model. Due to the stochastic nature of the Random Walk, we ran it 3× and averaged the results.

Table III summarizes the results in terms delta in accuracy and FLOPs reduction. This table clearly shows that our method outperforms random selection across the architectures. Importantly, although the results appear similar in terms of accuracy, Random Walk stops pruning prematurely. Specifically, on ResNet32/56, after removing 55.83% and 69.64% of FLOPs, Random Walk starts degrading accuracy. In particular, when the models become deeper, it is unable to remove more than 70% without degrading accuracy. Our method, on the other hand, successfully keeps removing more structures.

Overall, this experiment corroborates that carefully choosing the type of structure to eliminate, rather than selecting randomly, allows us to achieve significant FLOP reduction, as the pruning proceed further without compromising accuracy.

Do High-capacity Models Favor Certain Structures? In this experiment, we aim to determine if model capacity influences the preference for removing specific structures. Particularly, analyzing such a behavior is important to assess if our method is not converging to a trivial solution: remove one structure until complete and then start eliminating the other one. In order to verify this, we observe the initial pruning iterations.

On shallow architectures, ResNet32/44, our method tends to remove more layer than filter. For example, with 10 iterations of pruning (i.e., $K = 10$ in Algorithm 1), we observe a ratio of layers and filters removed of 7/3, on both architectures, where the first and second values indicate the number of occurrences our method eliminates layer and filters, respectively.

Building upon the above analysis, we extend our observations to deeper models, including ResNet56/110. For easy of exposition, we denote algorithm decisions as L (Layer) and F (Filter), and represent i and j consecutive layer and filter decisions as L^i and F^j, respectively. In the initial iterations on these deep and high-capacity models, a clear pattern emerges. The pattern on ResNet56 does not persist for long, it alternates between filters and layers for 6 iterations. On ResNet110, the pattern is more consistent, following the decision block (L, F^2) three times, then L^3, and repeating this sequence once more.

In general, previous experiments reveal a transition in preference: *the shallower the architecture, the greater the tendency to choose layers, while deeper models tend to prefer filters. But regardless of the architecture none structure is prevalent.*

The Role of Fine-tuning. Throughout our analysis, we note that without fine-tuning, our method favors eliminating layers instead of filters. It turns out that the internal representations become inconsistent due to variations in the magnitude of the weights. Previous works corroborate this behavior [10], [23], [25], where the authors argue that after pruning, subnetworks lose capacity. Fortunately, only a few epochs of fine-tuning are enough to restore subnetworks to an accuracy close to, or even higher than, their parent. Particularly, Manson el al. [23] also highlight that if a pruned network does not recover within one epoch, it suggests the network is *disconnected* from the loss manifold. For this reason, in both works [23], [25], the authors apply a 10-epoch fine-tuning before continuing the process.

Building upon the prior discussion, performing fine-tuning before the comparison process is crucial for making a more informed choice in our approach, favoring no structure; thus, this step adheres to the principle of recovering the subnetwork and determining which one is more similar to its parent.

Invariance to Similarity Metrics. This experiment shows that our method is robust to similarity metrics, achieving positive results across various similarity metrics. These metrics assess the proximity between objects, variables, models, and so forth. In this work, we leverage a similarity metric to identify the pruned subnetwork most similar to the parent, and although we use CKA, our algorithm supports other similarity metrics. To illustrate this, we conduct experiments using Linear [34], Gaussian Stochastic and Wasserstein [35] metrics. According to previous works [34], [35], such metrics emerge as potential alternatives to CKA. Following our previous experiment, we indicate the algorithm decisions as L (Layer) and F (Filter) until the method achieve the subnetwork with highest FLOP reduction with a positive delta in the accuracy.

Using the Linear metric, our method selected the path F^4, L, F, achieving a 56.18% FLOP reduction. With the Gaussian Stochastic metric, our method pruned only layers with the path L^7, achieving 47.78% of FLOP reduction. Finally, employing the Wasserstein metric it selected the path L, F, L^2, F, L^2, resulting in a 52.5% FLOP reduction.

These results confirm that our method indicates robustness to different similarity metrics. Aiming to consistently choose the best structure to remove, we show that our method is similarity metric-agnostic, and we highlight that using CKA provides better outcomes. Importantly, CKA computation is more efficient than other metrics (due to space constraints, we refer to previous works [34], [35] for additional information).

Robustness to Adversarial Samples. In critical domains like medical diagnosis and finance, robustness to out-of-distribution (OOD) and adversarial samples is as crucial as generalization [21], [36]. In this context, previous works studied the impact of pruning on adversarial robustness and OOD generalization, supporting that pruning mechanisms enhance these metrics [21], [36]. In this experiment, we aim to assess the impact of multiple structure removal promoted by our technique on adversarial attacks and OOD generalization on standard benchmarks. To achieve this, we employ the CIFAR-C [37] and CIFAR-10.2 [38] datasets and FGSM attack to test our pruned models. Specifically, we achieve improvements in robustness even at high compression rates; e.g., in CIFAR-10.2 at 81% of FLOP reductions, there are no signs of performance degradation in ResNet110. Additionally, at a surprising compression rate of 91%, our method enhances robustness against the FGSM attack by 2.5%. Lastly, on CIFAR-C, all of our pruned models exhibit better robustness compared to the unpruned baseline, reaching 96% of computational reductions with a positive delta in accuracy of almost 4%. We observe a similar trend for other architectures such as ResNet56. The results reinforce that pruning enhances robustness and our pruned models remain applicable in critical scenarios.

GreenAI. Previous studies confirmed that modern models lead to high carbon emissions (CO_2) due to substantial computational power and energy consumption during both for training and deployment [3], [13]. Fortunately, the computational gains of our pruned models directly translate into lower CO_2 emissions. As a concrete example, on ResNet56, our pruned model with the highest FLOP reduction achieves a 76.47% reduction in CO_2 emission. Another milestone is on ResNet110 by accomplishing a 83.31% of CO_2 reduction. Importantly, these pruned models also improve the financial cost. In particular, we reduce 76% and 83.67% of these costs on ResNet56/110. We estimate these values using the Machine Learning Impact Calculator [13]. In summary, we provide solid results toward GreenAI, helping to reduce CO_2 emissions and making deep models more financially accessible.

V. Conclusions

Modern pruning approaches exhibit promising results and distinct computational benefits in removing neurons (filters) or layers composing a model. However, few efforts exist in pruning both structures at once. In this work, we fill this gap by introducing a novel approach capable of eliminating both structures at once. The central idea behind our method involves deciding which structure (neuron or layer, each one represented by a subnetwork) to eliminate during an iterative pruning process. Throughout this decision process, we confirm that choosing subnetworks that preserve the internal representation with its parent (the network that yields the subnetworks) is an effective strategy. While our modeling enables a naive solution – random decision – we demonstrate that employing similarity metrics, such as Centered Kernel Alignment (CKA), leads to significantly better results. In this direction, we show that our method is similarity metric-agnostic, achieving positive results with various similarity metrics.

Through experiments on standard benchmarks and architectures, we validate the effectiveness of our technique. Specifically, it surpasses state-of-the-art pruning techniques by a notable margin. In particular, at high levels of FLOP reduction, most methods face challenges in maintaining accuracy, whereas our approach either improves it or experiences only a minimal drop. Notably, we achieve milestones in FLOP reduction such as a 95.82% FLOP reduction on ResNet110. Such achievements comes from the fact that our method leverages the best of both layer and neuron pruning, while existing methods are confined to a single structure.

Apart from the previous benefits, our pruned models exhibit robustness against different adversarial attacks. Finally, our models also reduce financial costs associated with training and fine-tuning and poses an important step towards GreenAI by reducing up to 83% of carbon emissions and financial costs required for training and fine-tuning modern architectures.

In summary, given the call for efficient models, particularly in the age of foundation models, we believe our work paves the way for a new chapter in accelerating models through pruning.

Room for Improvement. Our work offers opportunities for refinement. In this sense, one could incorporate more pruning settings as additional branches in our modeling.

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.

References

[1] N. M. et al., "Artificial intelligence index report 2025," Tech. Rep., 2025.

[2] Y. B. et al., "International ai safety report," Tech. Rep., 2025.

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

[4] A. D. et al., "The llama 3 herd of models," *ArXiv*, 2024.

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

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

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

[8] J. K. et al., "Layermerge: Neural network depth compression through layer pruning and merging," *ICML*, 2024.

[9] S. Kornblith, M. Norouzi, H. Lee, and G. E. Hinton, "Similarity of neural network representations revisited," in *ICML*, 2019.

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

[11] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, and J. M. Álvarez, "Structural pruning via latency-saliency knapsack," in *NeurIPS*, 2022.

[12] A. Ganjdanesh, S. Gao, and H. Huang, "Jointly training and pruning cnns via learnable agent guidance and alignment," in *CVPR*, 2024.

[13] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," in *NeurIPS*, 2019.

[14] S. G. et al., "Bilevelpruning: Unified dynamic and static channel pruning for convolutional neural networks," in *CVPR*, 2024.

[15] X. W. et al., "Auto-train-once: Controller network guided automatic network pruning from scratch," in *CVPR*, 2024.

[16] Y. Yang, Z. Cao, and H. Zhao, "Laco: Large language model pruning via layer collapse," *EMNLP*, 2024.

[17] A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. Roberts, "The unreasonable ineffectiveness of the deeper layers," in *ICLR*, 2025.

[18] B.-K. K. et al., "Shortened LLaMA: A simple depth pruning for large language models," in *ICLRW*, 2024.

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

[20] A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *ALT*, 2005.

[21] A. Jordao, G. C. de Araújo, H. de Almeida Maia, and H. Pedrini, "When layers play the lottery, all tickets win at initialization," in *ICCV*, 2023.

[22] J.-H. Luo and J. Wu, "Neural network pruning with residual-connections and limited-data," in *CVPR*, 2020.

[23] G. M.-W. et al., "What makes a good prune? maximal unstructured pruning for maximal cosine similarity," 2024.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[25] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," *ICLR*, 2024.

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

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

[28] Y. Zhang, M. Lin, C. Lin, J. Chen, and et al., "Carrying out CNN channel pruning in a white box," *TNNLS*, 2023.

[29] M. L. et al., "Pruning networks with cross-layer ranking & k-reciprocal nearest filters," *TNNLS*, 2023.

[30] Y. G. et al., "DAIS: automatic channel pruning via differentiable annealing indicator search," *TPAMI*, 2023.

[31] D. J. et al., "On the channel pruning using graph convolution network for convolutional neural network acceleration," in *IJCAI*, 2022.

[32] M. Alwani, Y. Wang, and V. Madhavan, "DECORE: deep compression with reinforcement learning," in *CVPR*, 2022.

[33] S. Zhong, G. Zhang, N. Huang, and S. Xu, "Revisit kernel pruning with lottery regulated grouped convolutions," in *ICLR*, 2022.

[34] A. H. Williams, E. Kunz, S. Kornblith, and S. W. derman, "Generalized shape metrics on neural representations," in *NeurIPS*, 2021.

[35] L. R. Duong, J. Zhou, J. Nassar, J. Berman, J. Olieslagers, and A. H. Williams, "Representational dissimilarity metric spaces for stochastic neural networks," *ICLR*, 2023.

[36] A. Bair, H. Yin, M. Shen, P. Molchanov, and J. M. Alvarez, "Adaptive sharpness-aware pruning for robust sparse networks," in *ICLR*, 2024.

[37] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *ICLR*, 2019.

[38] S. L. et al., "Harder or different? a closer look at distribution shift in dataset reproduction," in *ICML*, 2020.

[39] G. Liu, K. Zhang, and M. Lv, "SOKS: automatic searching of the optimal kernel shapes for stripe-wise network pruning," *TPAMI*, 2023.

[40] J. Sena, J. Barreto, C. Caetano, G. Cramer, and W. R. Schwartz, "Human activity recognition based on smartphone and wearable sensors using multiscale dcnn ensemble," *Neurocomputing*, 2021.

VI. Appendix

A. Robustness to Pruning Criteria

In this experiment, we highlight the potential of our method by illustrating its flexibility w.r.t popular pruning criteria c. Here, we prune ResNet32 using ℓ_1-norm (L1) and compare it with the Kullback-Leibler (KL). While preserving generalization performance, L1 achieves a FLOP reduction of 54.60%, and KL achieves 61.99%. From these results, we confirm that our method successfully works with others criteria.

B. Results on Shallow Architectures

In this experiment, we demonstrate the effectiveness of our method on shallow architectures, specifically ResNet32/44. Table IV summarizes the results. From Table IV, we highlight that the our method reduces 61.99% and 71.31% of FLOPs respectively from architectures ResNet32/44, without degrading the model accuracy. On ResNet32, we achieve a milestone of 96.15% FLOPs reduction, surpassing any other reports from previous work [6]. This behavior aligns with our analysis of deeper models and reinforces the effectiveness of our method on shallow architectures.

TABLE IV
Comparison of state-of-the-art methods on CIFAR-10.

	Method	Δ Acc.	FLOPs (%)
ResNet32	DAIS [30] (TNNLS, 2023)	(+) 0.57	53.90
	SOKS [39] (TNNLS, 2023)	(-) 0.80	54.58
	Pons et al. [7]	(+) 0.05	54.61
	Ours	**(+) 0.71**	**57.40**
	Pons et al. [7]	(-) 0.18	61.44
	Ours	**(+) 0.09**	**61.99**
	Ours	(-) 2.83	87.06
	Ours	(-) 7.63	**96.15**
ResNet44	Pons et al. [7]	(+) 0.22	62.95
	Ours	**(+) 0.66**	**71.31**
	Ours	(-) 1.26	**83.19**

Overall, our method accomplishes notable results, outperforming state-of-the-art pruning methods. As a clarification, the table shows few methods due to the absence of pruning studies reporting results on these architectures [5], [6].

C. Results on the Transformer Architecture

This experiment focuses on demonstrating the potential of our method beyond ResNet architectures. As prior studies indicate, pruning techniques extend to modern self-attention architectures, such as Transformers [5]. In this work, we show that our method is compatible with this architecture. To evaluate its efficiency, we alternate structural pruning between layers and heads in the multi-head attention blocks. We conduct these experiments using tabular data [40].

Our original, unpruned, Transformer architecture consists of 10 layers, each featuring 128 heads with projection dimensions of 64. We train this architecture for 100 epochs on each dataset before pruning it using the proposed method.

Figure 2 shows the results, where each blue circle represent a pruned model and an improvement in accuracy. According to the figure, all subnetworks show gains in accuracy. We intend to explore this behavior further in future research on LLMs.

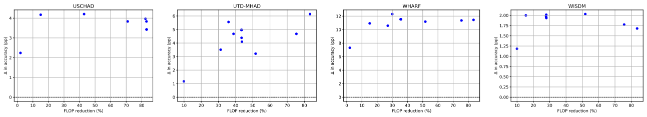

Fig. 2. Performance of our method pruning Transformer architectures.

We also demonstrate that our method, when pruning Transformers, is effective even when changing the pruning criterion. As a concrete example, we prune the model using KL and L1, achieving respectively 83.91% and 85.08% of FLOP reduction, while maintaining generalization.

Overall, the findings of this experiment shows a breakthrough compared to structural pruning on ResNet architectures and Figure 2 confirms that our method applies effectively to Transformers for tabular data.