Gustavo Henrique do Nascimento

Pruning Everything: Towards Efficient Deep Models Removing Any Structure

São Paulo

2026

Gustavo Henrique do Nascimento

Pruning Everything: Towards Efficient Deep Models Removing Any Structure

Original Version

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

Advisor: Prof. Dr. Artur Jordão Lima Correia

São Paulo

2026

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Ficha Catalográfica

do Nascimento, Gustavo Henrique
Pruning Everything: Towards Efficient Deep Models Removing Any Structure / G. H. do Nascimento -- São Paulo, 2026.
68 p.

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

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

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

Nome: Gustavo Henrique Do Nascimento

Pruning Everything: Towards Efficient Deep Models Removing Any Structure

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Aprovado em:

Banca Examinadora:

Prof. Dr(a).:

Instituição:

Julgamento: _____

Prof. Dr(a).:

Instituição:

Julgamento: _____

Prof. Dr(a).:

Instituição:

Julgamento: _____

Acknowledgements

I dedicate this work to all the people who supported me throughout this journey. Getting a master's degree was always one of my life goals, and completing it while working full time was a beautiful challenge, dividing my time between my passion for research and robotics. Even though my inner drive pushed me further, their encouragement, patience, and support helped me go beyond and achieve this accomplishment.

I would like to thank my family for their unconditional support. First and foremost, my parents, Ilson and Zilah, it would not have been possible to accomplish this without their support and encouragement to grow academically and professionally. I would also like to thank my brother, sisters, and little cousin, Camilla, Guilherme, Carol, and Otávio. This journey was not easy, and their support was extremely important in facing any barrier I had to break or goal that I had to achieve, and I would like to share this accomplishment with you. I would also like to thank Michele for supporting and believing in me throughout every sprint, challenge, and process that I had to face. Your support means a lot to me, and I am grateful for that.

I would also like to thank my research colleagues: Leandro, Ian, Keith, Guilherme, Carolina, and Bruno. Even though I could not participate in all the group meetings, thank you all for the exchange of ideas, insights, conversations, and hard work that we put together.

This work would not have been possible without the support of Prof. Dra. Anna Reali for first accepting me into the program and being my first mentor; thanks for all the meetings and teaching. This work would not have been possible without Prof. Dr. Artur Jordão, and I would like to have a separate paragraph to express my gratitude.

I am immensely grateful to my supervisor, Prof. Dr. Artur Jordão, for accepting this adventure of changing the entire course of my research to a whole new theme. His teaching made me a better researcher; thanks for guiding me through this path and improving my vision of science, and together we achieved something really important for the state-of-the-art.

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).

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. In this dissertation, we propose a new modeling 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, in our approach, we decide 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. Importantly, our method extends successfully from classic residual networks to modern architectures, including language and multimodal models. Throughout extensive experiments on vision, language, and multimodal benchmarks show consistent improvements and 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. At a 30% compression rate, our method exhibits drops of only 7.67 and 12.57 percentage points on LLaMA-2 7B and LLaMA-3 8B, respectively. Notably, this result at LLaMA-3 8B is approximately 6 percentage points higher than the state-of-the-art. Regarding the multimodal architecture, we achieve effective results across different compression rates. 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%. Finally, we deploy a pruned model on an extremely resource-constrained device, the ESP32. Our findings suggest that the proposed method provides a flexible, structure-agnostic, and efficient pruning approach. Code is available on GitHub.

Keywords: Pruning; Model Compression; Centered Kernel Alignment; GreenAI.

Resumo

O aprendizado profundo consolidou-se como um paradigma para a resolução de tarefas cognitivas. Contudo, o aumento da complexidade dos problemas torna os modelos mais profundos e computacionalmente custosos, dificultando sua aplicação em ambientes com recursos limitados. Estudos demonstram que a poda estrutural desses modelos reduz eficientemente sua complexidade e melhora a eficiência computacional. Estratégias bem-sucedidas nesse contexto incluem a remoção de neurônios ou de camadas, mas não de ambos simultaneamente. Portanto, a poda simultânea de diferentes estruturas permanece um problema em aberto. Com o objetivo de preencher essa lacuna e explorar os benefícios da remoção simultânea de neurônios e camadas, nesta dissertação propomos uma nova abordagem capaz de podar diferentes estruturas dentro de um mesmo modelo. Dadas duas subredes candidatas, uma resultante da poda de camadas e outra da poda de neurônios, nossa abordagem decide qual delas selecionar escolhendo aquela que apresenta a maior similaridade de representação em relação à sua rede de origem, utilizando a métrica *Centered Kernel Alignment* (CKA). A repetição iterativa desse processo resulta em modelos altamente esparsos que preservam a capacidade preditiva original. É importante destacar que nosso método se estende com sucesso desde redes residuais clássicas até arquiteturas modernas, incluindo modelos de linguagem e multimodais. Por meio de experimentos em *benchmarks* de visão, linguagem e multimodalidade, demonstramos melhorias consistentes e desempenho superior ao de técnicas do estado da arte. Em altos níveis de redução de *Floating Point Operations* (FLOPs), a maioria dos métodos do estado da arte apresenta degradação da acurácia, enquanto nossa abordagem a melhora ou apresenta apenas uma redução pequena. Em particular, nas redes ResNet56 e ResNet110, alcançamos reduções de FLOPs de 86,37% e 95,82%, respectivamente. Com uma taxa de compressão de 30%, nosso método apresenta quedas de apenas $7,67$pp e $12,57$pp nos modelos LLaMA-2 7B e LLaMA-3 8B, respectivamente. Em particular, o resultado obtido com o LLaMA-3 8B é aproximadamente 6pp superior ao estado da arte. Em relação à arquitetura multimodal, alcançamos resultados eficazes em diferentes taxas de compressão. Além disso, nossos modelos podados apresentam robustez a amostras adversariais e fora da distribuição e representam um passo importante em direção à *Green AI*, reduzindo as emissões de carbono em até 83,31%. Por fim, implantamos um modelo podado em um dispositivo com recursos extremamente limitados, o ESP32. Nossos resultados sugerem que o método proposto oferece uma abordagem de poda flexível, agnóstico a estrutura e eficiente. De modo geral, acreditamos que nosso trabalho abre um novo capítulo no campo da poda de redes neurais. O código está disponível no GitHub.

Palavras-chave: Poda; Compressão de Modelo; Centered Kernel Alignment; GreenAI.

List of Figures

Figure 1 – A binary decision tree of potential pruning outcomes and the paths . . 15
Figure 2 – Comparison with state-of-the-art techniques on the standard ResNet56 18
Figure 3 – Comparison between a plain network and a skip connection network . . 22
Figure 4 – Representation of a Large Visual-Language Model 28
Figure 5 – Different families of pruning techniques 30
Figure 6 – Proposed method at starting point 39
Figure 7 – Performance of our work on Transformer architecture for sensor data . 49
Figure 8 – Performance on Transformer architecture with different pruning criteria c 49
Figure 9 – Speed-up of the pruned models of ResNet 32/44/56/110 55

List of Tables

Table 1 – Comparison of Pruning Approaches 36
Table 2 – Comparison with state-of-the-art methods on CIFAR-10 43
Table 3 – Comparison with state-of-the-art pruning methods on ImageNet 44
Table 4 – Comparison with state-of-the-art methods on LLMs 45
Table 5 – Results on LLaVA-1.5 7B with different compression rates 46
Table 6 – Comparison of state-of-the-art methods on CIFAR-10 using ResNet32/44 47
Table 7 – Results on CIFAR-10 using ResNet32 with different criteria c 50
Table 8 – Comparison with random selection baseline (Random Walk) on CIFAR-10 51
Table 9 – Comparison with other similarity metrics on CIFAR-10 53

List of abbreviations and acronyms

CKA	Centered Kernel Alignment
CNN	Convolutional Neural Network
FLOPs	Floating Point Operations
LLM	Large Language Model
LVLM	Large Vision–Language Model
pp	Percentage points
ViT	Vision Transformer

List of symbols

L	Depth
S	Importance of the Structure
x	Input
X	Input Dataset
Y	Input Labels
$\mathcal{R}$	Internal Representation
$\mathcal{F}$	Neural Network
K	Number of Iterations
$\overline{y}$	Output
p	Percentage
c	Pruning Criterion
D	Size of Dataset
$\mathcal{F}^{'}$	Subnetwork
$\mathcal{F}_{f}^{'}$	Subnetwork pruned by filters
$\mathcal{F}_{l}^{'}$	Subnetwork pruned by layers
θ	Weights
N	Width

Contents

1 INTRODUCTION 13
1.1 Motivation 14
1.2 Research Statement 17
1.3 Objectives 17
1.4 Contributions 17
1.5 Work Organization 20

2 BACKGROUND 21
2.1 Neural Networks 21
2.2 Residual Networks 22
2.3 Internal Representations 23
2.4 Similarity Metrics 23
2.5 Pruning 24
2.6 Transformers 25
2.7 Large Language Models 26
2.8 Large Vision-Language Models - Multimodals Networks 27

3 RELATED WORK 30
3.1 Unstructured Pruning 31
3.2 Structured Pruning 32
3.3 Pattern Pruning 34
3.4 Multi-Structure Pruning 35
3.5 *Where are we?* - Overview 35

4 METHODOLOGY 37
4.1 Problem Statement 37
4.2 Definitions 37
4.3 Proposed Method 38

5 EXPERIMENTS 41
5.1 Experimental Setup 41
5.2 Comparison with State-of-The-Art 42
5.3 Comparison with State-of-The-Art in Large Language Models . . . 44
5.4 Effectiveness in Multimodal Architectures 46
5.5 Effectiveness in Shallow Architectures 46
5.6 Results on the Transformer Architecture for Tabular Data 48

5.7 Robustness to Pruning Criteria 49
5.8 Is CKA Better than Random Choice? 50
5.9 Do High-capacity Models Favor Certain Structures? 51
5.10 The Role of Fine-tuning 52
5.11 Invariance to Similarity Metrics 52
5.12 Robustness to Adversarial Samples 53
5.13 GreenAI 54
5.14 Deployment on Low-Cost Devices 54

6 CONCLUSIONS 56
6.1 Future Work 57

BIBLIOGRAPHY 59

1 Introduction

Deep learning drives unprecedented progress in various cognitive tasks and serves as the powerhouse for learning patterns from data (BENGIO et al., 2025; MASLEJ et al., 2024; MASLEJ et al., 2025). Multi-Layer Perceptrons, Convolutional Neural Networks (CNNs), Transformers, and Large Language Models (LLMs) represent a diverse set of deep learning architectures commonly applied to such tasks. Apart from these architectures, recent advances focus on increasingly large and general-purpose models (BENGIO et al., 2025; MASLEJ et al., 2025). Within this sphere, Bommasani et al. (BOMMASANI et al., 2021) introduced the term *foundation models* to describe models trained on large web-scale datasets. As the etymology of the term suggests, these models act as a foundational starting point for learning new, unseen, downstream applications.

Even though foundation and large models achieve impressive results, their performance incurs a significant computational cost, consequently leading to substantial environmental impacts during training (FAIZ et al., 2024; MORRISON et al., 2025). The scale of modern LLMs highlights the growing computational demands of contemporary AI systems. Recent model families span an unprecedented range of sizes, from 0.6B to 235B parameters in Qwen3 (YANG et al., 2025a) to 405B parameters in LLaMA 3 (GRATTAFIORI et al., 2024b), while DeepSeek-R1 reaches 671B parameters (DEEPSEEK-AI et al., 2025). Such scale requires massive computational infrastructure. In particular, training LLaMA 3 involves up to 16K NVIDIA H100 GPUs distributed globally, whereas inference with the largest 405B model requires 16 GPUs operating at 16-bit precision (GRATTAFIORI et al., 2024b). These examples illustrate the substantial computational resources required by modern models and reinforce the need for efficient model compression techniques.

Moreover, due to the infrastructure required to train and develop large models, only major companies have the resources to produce and offer them. A concrete example appears in the Artificial Intelligence Index Report (MASLEJ et al., 2025), where the authors estimate the training costs in dollars of GPT-4 and Gemini Ultra at approximately $79 million and $129 million, respectively. Besides the financial impact of training such costly models, their carbon emissions are also extremely high. As an illustration, the training of the previously mentioned models produced approximately 5184 and 8930 tons of CO_2, respectively (MASLEJ et al., 2025).

The aforementioned question highlights the challenge of deploying large models to real-world problems. To address these challenges, compression techniques pose an effective mechanism to reduce the size and latency of the models. Methods such as quantization (FRANTAR et al., 2023) and pruning (CHENG; ZHANG; SHI, 2024b) focus

on reducing the number of parameters, which consequently lowers computational cost, reduce training costs and carbon emission, working towards GreenAI.

Efforts regarding compression of deep models confirm the effectiveness of pruning methods in reducing computational demands without degrading the predictive ability of models (CHENG; ZHANG; SHI, 2024b; TAVARES et al., 2025). Most pruning methods remove small structures composing a model such as weights or filters (CHENG; ZHANG; SHI, 2024b; HE; XIAO, 2023). 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 (PONS et al., 2024; KIM et al., 2024b). 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). For example, Figure 1 illustrates a binary decision tree that outlines the possible pruning paths. Throughout this work, references to Figure 1 use path to the right to represent layer pruning, while path to the left refers to filter pruning. Therefore, in Figure 1, it is possible to illustrate the behavior previous mentioned about the limitation of layer pruning, 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).

1.1 Motivation

The rapid evolution of Convolutional Neural Networks, Large Language Models, and Large Vision-Language Models improves their ability to understand, reason over, and generate multimodal content (BROWN et al., 2020; OpenAI, 2023; LIU et al., 2023a). However, these advances also increase model size, computational cost, memory requirements, and energy consumption, limiting deployment in resource-constrained environments (STRUBELL; GANESH; MCCALLUM, 2019; PATTERSON et al., 2021; HOFFMANN et al., 2022). Model compression emerges as a fundamental research direction for improving the efficiency of modern deep neural networks while preserving predictive performance (TAVARES et al., 2025; CHENG; ZHANG; SHI, 2024b).

As a concrete example that motivates this work, Pons et al. (PONS et al., 2024) 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. Technically speaking, existing filter-pruning approaches share similar limitations. Considering that the literature focuses on eliminating individual structures, filters, or layers but not both, a naive strategy for obtaining high-level compression is to eliminate different structures throughout a pruning process (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b). Furthermore, achieving a high compression rate, allows deployment on low-cost devices and make cheaper the

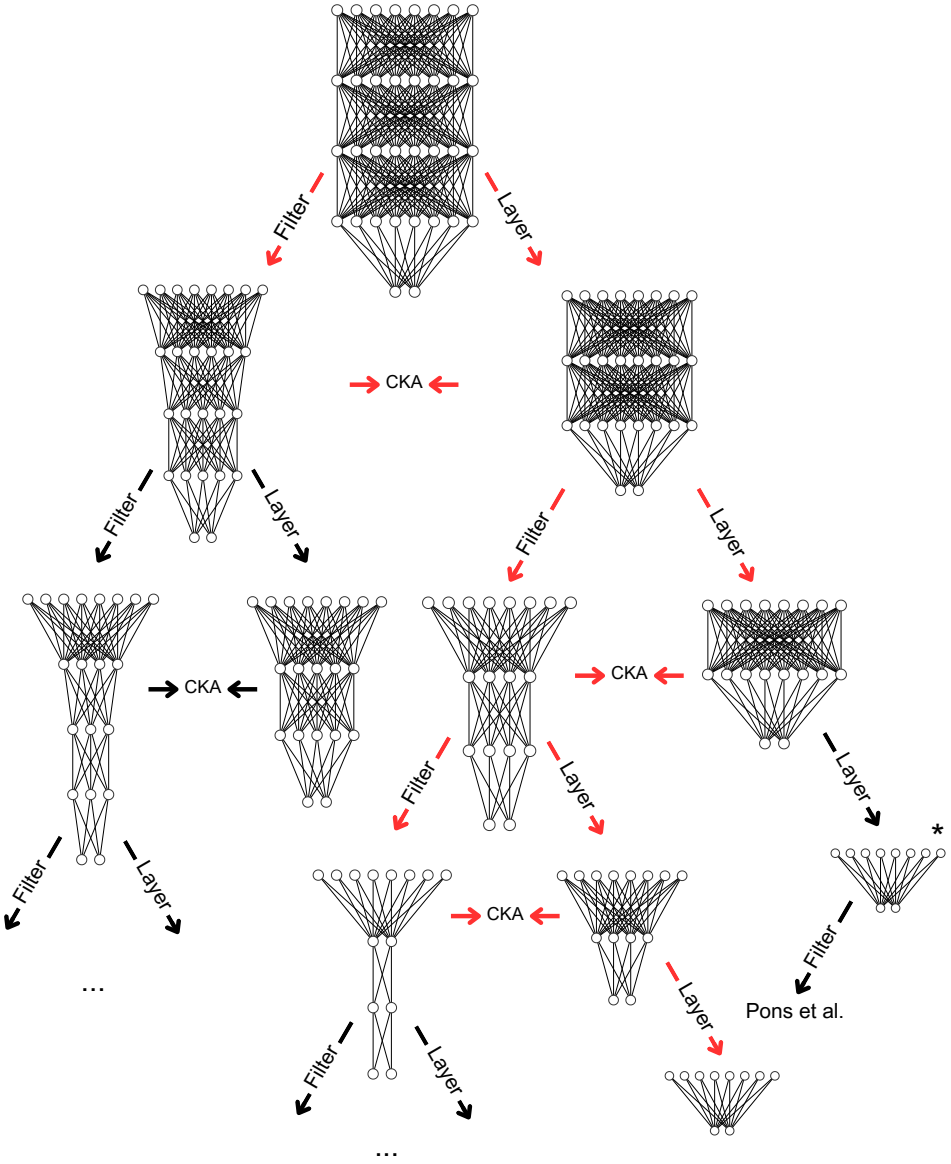


Figure 1 – A binary decision tree of potential pruning outcomes and the paths (red arrows) to follow using our approach. The process starts with an unpruned deep network (parent), for which we generate two subnetworks (pruned): one from layer pruning and the other from filter pruning. Then, with a similarity metric (i.e., Centered 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.

implementation of efficient neural networks.

From the previous discussion, a possible approach to overcome the limitations of existing structured pruning involves alternating between structures. In this direction, a natural question arises:

> *How do we decide which structure to remove during pruning?* Figure 1 allows restructuring the question as follows: *given a network (the parent[1]), 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 Centered Kernel Alignment (CKA) (KORNBLITH et al., 2019) 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, 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 (PONS et al., 2024; MURALIDHARAN et al., 2024; SHEN et al., 2022b; GANJDANESH; GAO; HUANG, 2024), we perform this process iteratively and, for each iteration, our method chooses the candidate greedily. This means it selects the locally proper candidate at each pruning iteration (one level in Figure 1) and ensures computational efficiency with $O(log(n))$ complexity, where n is the set of all possible candidate subnetworks.

Overall, although pruning strategies are effective, current methods show limited progress due to inherent trade-offs between model accuracy, inference latency, FLOPs reduction, and computational cost. For example, existing structural pruning methods primarily optimize individual architectural components, leaving the joint optimization of multiple structures largely unexplored. To address this limitation, this dissertation proposes a multi-structure pruning approach that simultaneously removes different structural components, including attention heads and layers, to exploit structural redundancy more effectively. In the following chapters, we discuss prior work that largely overlooks structured pruning. Most approaches focus on removing a single type of structure, such as filters or layers. In this context, we introduce a new pruning technique that opens a potential research direction within the structured pruning domain.

[1] Following Mason et al. (MASON-WILLIAMS; DAHLQVIST, 2024), we use the term parent referring to a network that generates the subnetworks (i.e., pruned models).

1.2 Research Statement

Throughout this work, we confirm 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).

1.3 Objectives

From a theoretical perspective, our goal is to demonstrate that, in an iterative pruning process, similarity metrics can guide the selection of subnetworks that remain closer to their parent models. By preserving this representational proximity, each subnetwork better maintains the generalization capabilities of the original model throughout the pruning iterations, enabling high compression rates without significantly compromising performance.

From a practical perspective, our goal is to achieve significant computational gains by removing multiple types of structures, rather than limiting the pruning process to a single one. We aim to achieve substantial FLOP reductions that lead to efficient models. Consequently, by producing lighter architectures, we also target lower inference costs and reduced carbon emission and other associated costs. Finally, through high compression rates, we seek to make these models more accessible to the research community and suitable for deployment on low-cost devices.

1.4 Contributions

Among our contributions, we highlight the following. We introduce a new form of pruning that eliminates both filters and layers from deep models and leverages the best of existing techniques. From a practical perspective, our method contributes to achieving a new milestone in computational gains. Our modeling allows to not confine the removal of only one structural type and integrates the benefits of both the filter and the 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 2 introduces one of our contributions 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.

Overall, we summarize the main contributions as follows.

- We propose a new form of pruning able to achieve high levels of compression. We confirm that using a similarity metric is more efficient than naively choosing between the types of structure.
- Extensive experiments across a broad range of benchmarks and architectures confirm our work promotes models that outperform state-of-the-art pruning techniques.

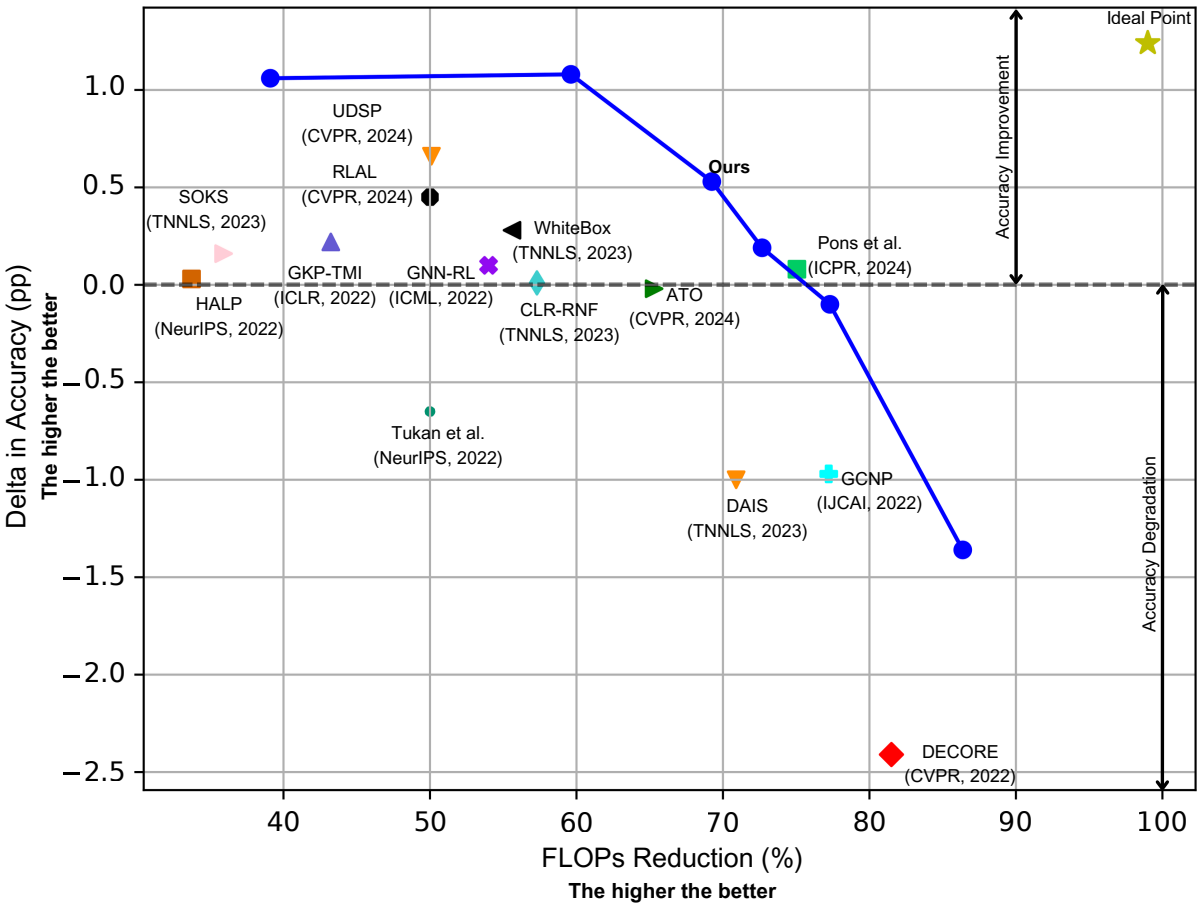


Figure 2 – Comparison with state-of-the-art techniques on the standard ResNet56 + CIFAR-10 setting. Our work significantly outperforms existing layer and filter pruning techniques by a large margin. Specifically, compared to other pruning approaches, our work enable us s to remove over 80% of FLOPs (Floating 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 work achieves better trade-offs in accuracy and computational efficiency (measured by FLOPs). These results confirm how our contributions enable 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.)

Specifically, we surpass existing methods by achieving 61.99% and 71.31% FLOPs reduction on popular architectures while increasing accuracy. On deeper architectures, we obtain a milestone of 86.37% and 95.82% FLOPs reduction on ResNet56 and ResNet110.

- We confirm the effectiveness of our method on modern architectures, including large language models and multimodal models. For LLMs, our method outperforms most state-of-the-art approaches. In multimodal architectures, to the best of our knowledge, this work is one of the few to explore pruning in this setting, establishing a baseline and opening new directions for future research.
- In terms of GreenAI (LACOSTE et al., 2019; FAIZ et al., 2024; MORRISON et al., 2025), our 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 (GRATTAFIORI et al., 2024a; BENGIO et al., 2025) and the increasing need for efficient models, we believe our contributions open a new chapter for achieving higher levels of compression by breaking down the barriers of existing strategies that rely solely on one structure.

As a result of these contributions, we have published the following paper during this research:

1. **Gustavo Henrique do Nascimento**, Ian Pons, Anna Helena Costa Reali, Artur Jordão. *Pruning Everything, Everywhere, All at Once*, International Joint Conference on Neural Networks (IJCNN), 2025.
2. Carolina Tavares, Leandro Mugnaini, **Gustavo Henrique do Nascimento**, Ian Pons, Keith Ogawa, Guilherme Stern, Lucas Libanio, Aline Paes, Anna Helena Reali Costa and Artur Jordão *Tiny Titans: Efficient Large Vision, Language and Multimodal Models through Pruning*, Conference on Graphics, Patterns and Images (SIBGRAPI), Tutorial Track, 2025.
3. **Gustavo Henrique do Nascimento**, Keith Ogawa Ando, Ian Pons, Anna Helena Costa Reali, Artur Jordão. *Pruning Everything: A Multi-Structure Pruning for Scalable Model Compression*, Neural Computing and Applications, Springer, 2026 (**Under review**).

Finally, to ensure replicability of this work, we make the source code and trained model publicly available at: https://github.com/NascimentoG/PruningEverything. The pruned LLMs are available at: https://huggingface.co/NascimentoGH/PruningEverything.

1.5 Work Organization

We organize the remainder of this dissertation as follows. In Chapter 2, we give the essential background to comprehend this work. In Chapter 3, we review the literature on pruning techniques related to our work. In Chapter 4, we define our problem statement and the proposed method. In Chapter 5, we show and discuss the experimental results. In Chapter 6, we summarize our findings and present the conclusions of this dissertation and directions for future work.

2 Background

To support a better understanding of this dissertation, we first introduce foundational concepts involving neural networks and pruning. We begin by explaining what neural networks are and highlight the key characteristics that enable removing structures from these models (pruning). This basics also requires an overview of residual networks, similarity metrics, and pruning strategies.

2.1 Neural Networks

Let $\mathcal{F}(\cdot, \theta)$ represents a neural network parameterized by a set of weights θ. Given an input $\mathbf{x}$, the network produces a prediction $\overline{y}$ based on its weights. Equation 2.1 shows the output:

$$\overline{y} = \mathcal{F}(x, \theta). \tag{2.1}$$

Formally, a neural network consists of a sequence of non-linear transformations f in terms of:

$$\mathcal{F}(x, \theta) = \overline{y} = f_L(\ldots, f_2(f_1(x, \theta_1), \theta_2), \theta_L), \tag{2.2}$$

where L denote the depth of the neural network (i.e., the number of layers). In addition to depth, another key architectural aspect is width, denoted by N, which refers to the number of neurons per layer and also influences the total number of parameters.

Follow by these definitions, one of the key factors behind of neural networks is the ability to scale their depth (i.e., grow in L). Szegedy et al. (SZEGEDY et al., 2015) reinforces the previous statement with the paradigm *"We need to go deeper"*. From a certain perspective, to *"go deeper"* allows the network to extract increasingly complex features from data and tackle a broader range of tasks (SZEGEDY et al., 2015). However, as architectures grow and become more expressive, increasing depth often requires increasing width (N). This expansion leads the model into an overparameterized regime, where the number of trainable parameters exceeds the number of available training samples. Despite this high capacity, models with a large number of parameters are more likely to memorize the training data rather than learn generalizable patterns.

Subsequently, another important feature to the neural network is plasticity. In neural networks, it refers to the ability to adapt its internal parameters (i.e., weights and biases) to recover after an alteration on its structure (MITTAL et al., 2018). Moreover,

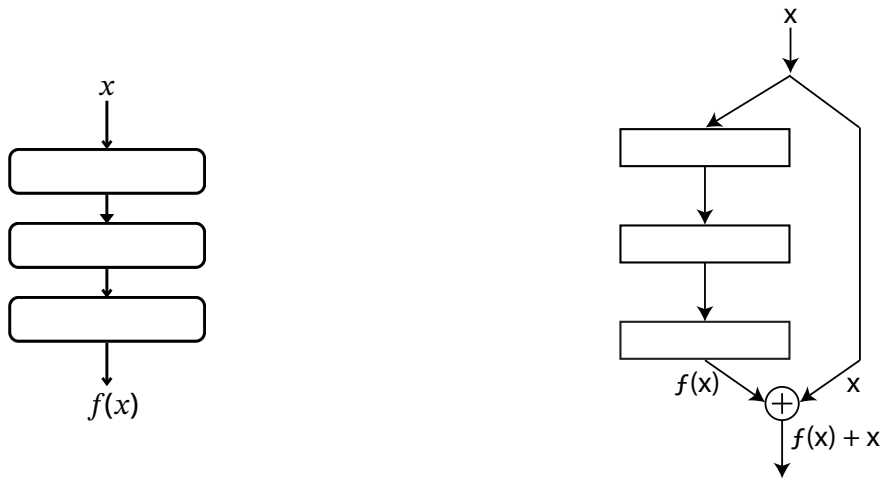
if training continues on new data, the network may suffer from catastrophic forgetting, where learning new tasks interferes with or overwrites previously acquired knowledge.

These two aspects drive our motivation to explore pruning. Overparameterized models include unnecessary and redundant structures, and we can remove them without significantly harming generalization. After pruning, the network loses connectivity and requires retraining to regain performance. At this point, plasticity becomes crucial: by adjusting its internal parameters, the model not only recovers but sometimes even improves.

2.2 Residual Networks

As networks grow deeper, they become significantly more complex to train, requiring an enormous amount of data to reach satisfactory performance (HE et al., 2016). Following the idea that deeper models are able to improve more the predictive performance, the increasing of the model brings with itself the degradation problem. In this regard, He et al. (HE et al., 2016) introduce the degradation problem, where as the network increases in depth, it begins to expose this issue, the accuracy saturates and then degrades quickly. This degradation is not caused by overfitting, and the authors address this problem by introducing *deep residual learning*.

Figure 3 illustrates the concept of residual learning. Figure 3a represents a plain network, where the layer is followed by another layer and given the input x, the output is $f(x)$. On the other hand, the residual network introduces the *Skip Connection*, Figure 3b shows that the output of a Skip Connection network is $f(x) + x$. He et al. (HE et al., 2016) affirm that this feature mitigates the problem of vanishing gradients, where this phenomenon occurs when the weights become extremely small during backpropagation through many layers.



(a) Plain Network **(b)** Skip Connection Network

Figure 3 – Comparison between a plain network and a skip connection network.

Based on these properties, the authors introduce the Residual Network (ResNet), a neural network architecture composed of building blocks, and each one has it Skip Connection. In this dissertation, we focus on pruning some of these models, due to technical details behind this architecture. However, Skip Connections impose a technical constraint: the first and last layers must have the same number of filters to perform the $f(x) + x$ operation, hence there is the restriction of removing the first or last layer.

2.3 Internal Representations

Image-related tasks often adopt the Convolutional Neural Network (CNN) architecture due to its ability to efficiently capture spatial hierarchies in visual data (GEIRHOS et al., 2019). A typical CNN is one that contains spatially local connections and has patterns of weights that replicate across the units in each layer. It consists of a sequence of convolutional layers, and it is the key to hierarchical feature extraction, that processes the feature maps output by applying learnable filters (a.k.a kernels). The filter slides over the input via the convolution operation to extract local features (RUSSELL; NORVIG, 2021; GOODFELLOW; BENGIO; COURVILLE, 2016). These filters perform convolution operations that produce feature maps (a.k.a internal representations) (XIONG et al., 2024), encoding both spatial and semantic information. As the network goes deeper, these feature maps progressively capture more abstract and complex patterns. Overall, the architecture of CNNs allows the model to learn low-level features such as edges and textures in the first layers, while deeper layers produce maps that respond to more complex and task-specific patterns, such as parts of objects or semantic regions. This makes CNNs particularly powerful for tasks such as image classification, segmentation, and object detection. In CNNs, feature maps constitute the internal representations of the network, which the model gradually learns throughout the training process using the available data.

Technically speaking, feature maps are three-dimensional tensors whose dimensions correspond to the height, width, and number of filters (channels) applied in that layer. The number and configuration of these maps directly influence the representational capacity and computational complexity of the network. Consequently, internal representations serve as a key target for compression strategies such as structured pruning, where the process removes entire channels or filters. Guided by the contribution and redundancy of each feature map, some forms of pruning methods can reduce the size of the model and the inference time while preserving the performance of the task.

2.4 Similarity Metrics

A key component of our work involves using a similarity metric to guide the pruning decision process. A similarity metric is a mathematical function that measures how close

two inputs are (KLABUNDE et al., 2025). In the context of pruning, we calculate the similarity between the original neural network ($\mathcal{F}$) and the subnetwork ($\mathcal{F}^{'}$). Let $\mathcal{M}(\cdot,\cdot)$ denote a similarity function. For example, Equation 2.3 takes $\mathcal{F}$ and $\mathcal{F}^{'}$ as input and produces a similarity score.

$$score = \mathcal{M}(\mathcal{F}, \mathcal{F}^{'}). \tag{2.3}$$

In our approach, the similarity between $\mathcal{F}$ and $\mathcal{F}^{'}$ is determined from their internal representations, such as feature maps. These representations provide information about how the network processes its inputs and allow us to assess whether the subnetwork preserves the behavior of the original model. Overall, these metrics allow us to compare internal representations, such as activations or feature maps, across different layers or subnetworks. For example, one can apply a simple similarity metric like a Linear metric (WILLIAMS et al., 2021). Due to the importance of Centered Kernel Alignment (CKA) to our work, we reserve Section 4.2 to explain it in detail. In our experiments, we also explore other metrics, such as Gaussian and Wasserstein (DUONG et al., 2023).

2.5 Pruning

Despite the benefits of neural networks, they do not come without drawbacks. Problems that appear with recent architectures are due to their high computational costs inherent from complexity and size of the models, and by consequence, the carbon emissions resulting from energy consumption while training or making inferences (CHENG; ZHANG; SHI, 2024b; FAIZ et al., 2024). Among the existing solutions to overcome these problems, pruning emerges as an effective hardware-agnostic technique to improve model efficiency (CHENG; ZHANG; SHI, 2024b; LUO et al., 2025). This technique aims to identify and remove network components that contribute little to the predictive capabilities of the model. Therefore, the central idea behind this family of techniques is to remove structures from the architecture, such as weights, filters, or even entire layers. Using specific evaluation metrics (a.k.a pruning criteria), the method selects and discards redundant or noninformative structures, producing a smaller subnetwork that retains the core behavior of the original architecture.

Previous works categorize the type of pruning in unstructured, structured, and pattern pruning (CHENG; ZHANG; SHI, 2024b), and we explain this later in Chapter 3. In this dissertation, our work is on structured pruning.

Algorithm 1 summarizes the steps of a structured pruning process. In general, in each iteration of Algorithm 1, a pruning criterion c assesses the importance S of the structures in the network $\mathcal{F}$. The pruning criterion c is a function that receives, in this case, the individual structure (e.g. layers or filters) and yields a score, in this work we

Algorithm 1 Pruning Structures of Deep Networks

1: **Input:** Network $\mathcal{F}$, Pruning Criterion c, Pruning Ratio p
2: **Input:** Number of Iterations K
3: **Output:** Pruned Network $\mathcal{F}'$ (a.k.a Subnetwork)
4: **for** $k \leftarrow 1$ **to** K **do**
5: $S \leftarrow c(\mathcal{F})$ ▷ Set importance for each structure
6: $I \leftarrow p\%$ Unimportant structures based on S
7: $\mathcal{F}' \leftarrow \mathcal{F} \setminus I$ ▷ Removes the structures indexed by I
8: Update $\mathcal{F}'$
9: $\mathcal{F} \leftarrow \mathcal{F}'$
10: **end for**

explore different pruning criteria in Section 5.7. Taking advantage of this importance, the algorithm removes $p\%$ of the least important structures, resulting in a subnetwork $\mathcal{F}'$. The algorithm then adjusts the parameters (see line 8), in a fine-tuning process, and repeats the process until it reaches all K iterations. The result of a pruning iteration is a pruned model ready for inference or other applications.

2.6 Transformers

Transformers define a class of neural architectures designed to model sequential data through attention mechanisms. Vaswani et al. (VASWANI et al., 2017) introduced the Transformer architecture to replace recurrent and convolutional structures with a fully attention-based design, enabling more efficient parallelization and improved long-range dependency modeling.

The core component of a Transformer is the self-attention mechanism, allowing the model to weigh the relevance of each element in a sequence with respect to others (JURAFSKY; MARTIN, 2026). Given an input sequence, the model projects an input into queries, keys, and values, and computes attention scores to produce context-aware representations. This mechanism enables the model to capture global dependencies without relying on recurrence.

Transformers consist of a stack of layers composed of multi-head self-attention and position-wise feed-forward networks. Multi-head attention enables the model to capture information from different representation subspaces, while positional encodings inject token-order information into the architecture (JURAFSKY; MARTIN, 2026). Unlike a single attention head, multiple heads allow the model to learn diverse attention patterns simultaneously, increasing its representational capacity (VASWANI et al., 2017).

Beyond their expressive power, Transformers offer favorable computational properties. Self-attention connects all positions in a sequence through a constant number of sequential operations, enabling high parallelization and facilitating the learning of

long-range dependencies (VASWANI et al., 2017). Furthermore, individual attention heads often specialize in different syntactic and semantic relationships, introducing a degree of redundancy that motivates structured pruning (MICHEL; LEVY; NEUBIG, 2019). These characteristics make Transformers particularly well-suited to compression techniques targeting specific architectural components.

In this dissertation, we investigate multi-structure pruning in Transformers by jointly removing different structural components, including layers and attention heads. Our approach leverages the modular design of the architecture to reduce model complexity while preserving predictive performance.

2.7 Large Language Models

The Transformer architecture marked a fundamental shift from recurrent and convolutional architectures by introducing self-attention mechanisms that process all sequence positions in parallel (VASWANI et al., 2017). Its modular design facilitates structured pruning and model compression techniques (LIU et al., 2023; SUN et al., 2020) while serving as the foundation for modern Large Language Models (LLMs). These models leverage self-attention to capture complex statistical patterns in natural language and perform a wide range of tasks without requiring task-specific architectural modifications.

LLMs primarily rely on two pretraining paradigms: autoregressive modeling and masked language modeling. Autoregressive models, such as the GPT family (BROWN et al., 2020; OpenAI, 2023), predict the next token from the preceding context, enabling coherent left-to-right text generation by modeling the causal structure of language. In contrast, masked language models such as BERT (DEVLIN et al., 2019) reconstruct randomly masked tokens from bidirectional context and learn contextual representations that excel in discriminative tasks. Encoder-decoder architectures, including T5 (RAFFEL et al., 2020), unify these approaches by casting all tasks into a text-to-text framework, thereby improving model versatility across diverse applications.

These advances enable LLMs to perform a broad range of tasks, including text generation, question answering, summarization, machine translation, code generation, and complex reasoning (ZHAO et al., 2026; OpenAI, 2023). Their ability to adapt through prompting, often with only a few demonstrations, establishes LLMs as a dominant paradigm in modern artificial intelligence. In particular, in-context learning allows these models to solve previously unseen tasks without gradient-based parameter updates and indicates that scaling laws influence not only optimization behavior but also downstream task performance (KAPLAN et al., 2020; HOFFMANN et al., 2022).

Despite these advances, important challenges remain. LLMs can generate fluent but factually incorrect content, a phenomenon known as hallucination (JI et al., 2023),

and they may reproduce or amplify biases present in their training data (DIXON et al., 2018; BENDER et al., 2021). Moreover, training and deploying these models require substantial computational resources and raise concerns about environmental impact and equitable access to state-of-the-art systems (STRUBELL; GANESH; MCCALLUM, 2019; PATTERSON et al., 2021). Current research addresses these limitations through improved training strategies, retrieval-augmented generation (LEWIS et al., 2020), constitutional AI frameworks (BAI et al., 2022), and model compression techniques, particularly pruning, that reduce computational cost while preserving model performance (FRANTAR; ALISTARH, 2023).

Model compression emerges as a solution to the challenges associated with energy consumption and model scaling. Pruning represents a compelling approach because it enables model size reduction while limiting performance degradation and facilitating deployment on less powerful hardware. In this dissertation, we focus on autoregressive Transformer-based models because of their widespread use in generative artificial intelligence, including the LLaMA family (TOUVRON et al., 2023; GRATTAFIORI et al., 2024b), as well as their adoption as language backbones in multimodal generative models such as LLaVA (LIU et al., 2024). We investigate multi-structure pruning in Transformers by jointly removing different structural components, including layers and attention heads. As LLMs and Large Vision-Language Models rely on Transformer-based architectures, our approach can extend naturally to these models. We leverage the modular design of Transformers to reduce model complexity while preserving predictive performance.

2.8 Large Vision-Language Models - Multimodals Networks

Building on the success of Large Language Models, recent advances in artificial intelligence introduce Large Vision-Language Models (LVLMs) capable of jointly processing and integrating textual and visual information. In particular, these models learn shared representations that align text and images, enabling cross-modal understanding, reasoning, and generation. Given an image and a textual prompt, LVLMs jointly process both inputs to generate natural language responses grounded in visual evidence.

Most Large Vision-Language Models adopt a modular architecture that combines a specialized visual encoder with a pre-trained Large Language Model (LIU et al., 2023a). The visual encoder, typically a Vision Transformer (ViT) (DOSOVITSKIY et al., 2021), extracts high-level semantic features from raw images. Figure 4 illustrates the architecture of a LVLM. A projection layer, often referred to as a bridge, maps these visual features into the embedding space of the LLM, allowing the language model to process visual and textual tokens within a shared representation. This design enables the LLM to interpret visual features as a sequence of *visual words*, thereby grounding language generation in

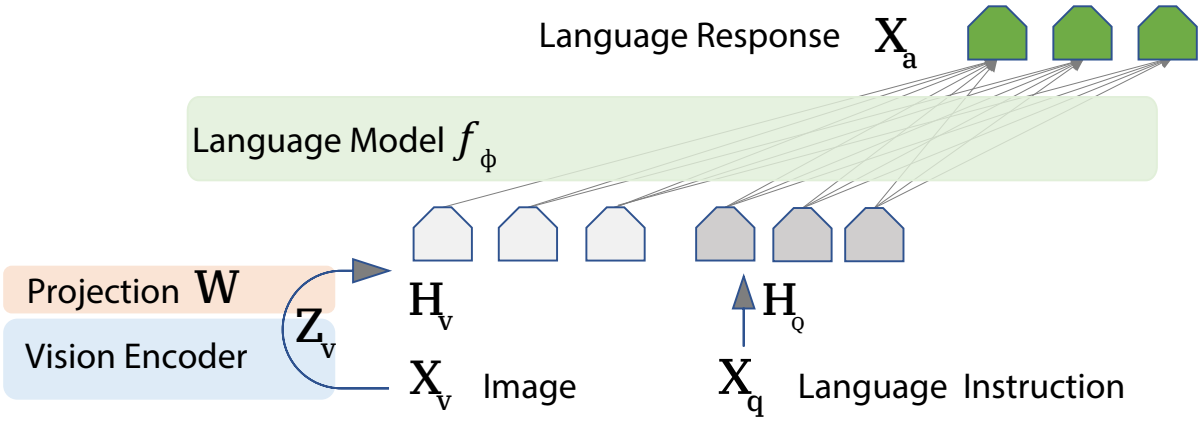


Figure 4 – Representation of a Large Visual-Language Model. Adaptated from (LIU et al., 2023a)

visual context.

Two alignment strategies pave the evolution of LVLMs: contrastive learning and generative alignment. Contrastive models, exemplified by CLIP (RADFORD et al., 2021), employ a dual-encoder architecture that maximizes the similarity between paired image-text samples while separating unpaired samples in a shared embedding space. This strategy enables strong zero-shot retrieval and classification performance but does not support autoregressive text generation. Generative models, such as Flamingo (ALAYRAC et al., 2022) and BLIP-2 (LI et al., 2023), address this limitation by introducing dedicated modules that connect visual representations to a pre-trained LLM. Flamingo employs a Perceiver Resampler to process variable-length visual inputs, whereas BLIP-2 introduces the Querying Transformer (Q-Former), which uses a set of learnable query tokens to extract the most relevant visual information before passing it to the LLM. This design efficiently transfers visual information to the LLM while preserving rich semantic representations.

The introduction of multimodal instruction tuning further advances Large Vision-Language Models from multimodal understanding to instruction-following and complex visual reasoning. LLaVA (LIU et al., 2023a) show that fine-tuning a vision-language model on instruction-following data generated with GPT-4 substantially improves conversational abilities and visual reasoning performance. These results suggest that high-quality alignment data plays a critical role in developing multimodal capabilities beyond what model scale alone can achieve.

Recent research shifts toward native multimodality. Unlike modular architectures that combine pre-trained visual encoders and language models separately, native multimodal models learn visual and textual representations jointly from interleaved multimodal data. This training strategy enables the model to capture cross-modal relationships throughout the learning process and improves the integration of visual and linguistic information. Native multimodal architectures also seek to reduce limitations observed in

modular systems, including visual hallucinations and weaker spatial reasoning capabilities. As these models continue to evolve, they establish new state-of-the-art results in tasks such as Visual Question Answering (VQA), image captioning, and grounded language understanding.

3 Related Work

Model compression comprehends finding a more efficient version of a given architecture while preserving predictive performance. Most successful mechanisms in this line of research rely on pruning (XU; MCAULEY, 2023; PONS et al., 2024; YANG; CAO; ZHAO, 2024; MURALIDHARAN et al., 2024; SUN et al., 2024). Recent literature (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b) categorizes this family of methods into two approaches: 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. To provide a clearer view of recent advances in this field, this chapter presents the state of the art on unstructured, structured, and pattern pruning. Figures 5 (a-d) show an example in an illustrative way of types of pruning, and Figure 5 (e) summarizes the timeline on the evolution of pruning.

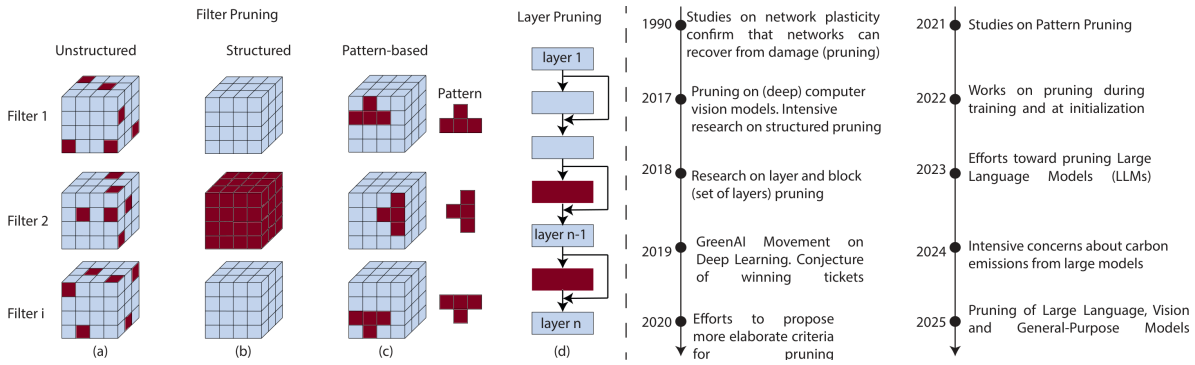


Figure 5 – Different families of pruning techniques. Blue and red areas indicate, respectively, unpruned and pruned structures (weights, filters or layers). Each cuboid (tensor) in (a)-(c) stands for a $k \times k$ filter. (a) Unstructured pruning removes weights at arbitrary locations, thus yielding 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.

.

3.1 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 (HAN et al., 2015).

In the pruning literature (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b), this mechanism is referred to as unstructured pruning because it removes weights at arbitrary locations (see Figure 5 (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 (ELSEN et al., 2020). However, recent NVIDIA GPUs, including the A100 and the more recent Hopper architecture (LUO et al., 2024), target sparse computing and improve its efficiency.

One of the pioneering works in unstructured pruning is the work by Han et al. (HAN et al., 2015), who focus on removing weights with low ℓ_1-norms. Although LeCun et al. (LECUN; DENKER; SOLLA, 1989) and Hassibi et al.(HASSIBI; STORK; WOLFF, 1993) had previously explored unstructured pruning, Han et al. (HAN et al., 2015) 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. (SUN et al., 2024) 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. (FRANTAR et al., 2023) 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* (FRANKLE; CARBIN, 2019). 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 (SHEN et al., 2022a; WANG et al., 2022).

Overall, unstructured pruning offers a simple approach to achieve higher compression rates without sacrificing accuracy. In particular, it can compress a ResNet50 by 5.96× with no accuracy loss (ZHOU et al., 2021). 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 (NIU et al., 2020; ELSEN et al., 2020). Therefore, due to those constrains, our work focus on another pruning approach.

3.2 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 5 (b)). Importantly, its structured nature enhances compatibility with existing hardware and software, facilitating practical acceleration improvements. 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) (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b).

To the best of our knowledge, Li et al. (LI et al., 2017) conducted the first successful structured pruning by extending the ℓ_1-norm criterion by Han et al. (HAN et al., 2015) to prune filters in CNNs rather than individual weights. Existing literature categorizes structured pruning according to the type of structure it targets (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b): filters or entire layers. Within the first category, existing studies treat filters and channels interchangeably (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b).

Building on the foundational work by Li et al. (LI et al., 2017), several researchers refined structured filter pruning by developing adaptive methods that guide pruning decisions more effectively (HE et al., 2020; LIU et al., 2021; LIN et al., 2020). For example, He et al. (HE et al., 2020) proposed adjusting pruning parameters layer by layer to better balance compression and accuracy. More recently, Gupta et al. (GUPTA et al., 2024) 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. Another concrete example, Gao et al. (GAO et al., 2024b) 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 (fixed, not changing during inference) filter pruning.

From the lens of layer pruning, 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 (VEIT; WILBER; BELONGIE, 2016; LUO et al., 2025).

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) (VEIT; WILBER; BELONGIE, 2016). Within this sphere of research, some works suggest that pruning blocks of layers is more effective than removing individual layers (CHEN; ZHAO, 2019; ZHOU; YEN; YI, 2022). 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. (ZHANG; BENGIO; SINGER, 2022) 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 (CHEN; ZHAO, 2019; ZHOU; YEN; YI, 2022; PONS et al., 2024). Also, Dror et al.(DROR et al., 2022) and Fu et al.(FU et al., 2022) employed structural reparameterization to merge layers, simplifying network depth and mitigating the issue of low variance in weight-based scores. Ganjdanesh et al. (GANJDANESH; GAO; HUANG, 2024) addressed the problem with reinforcement learning (RL), pruning iteratively while updating both RL and model parameters. Wu et al. (WU et al., 2024) proposed a method for pruning while working alongside a controller network to avoid falling into local optima. Kim et al. (KIM et al., 2024b) compressed models by using a layer merge method that removes layers and activation functions. Yang et al. (YANG; CAO; ZHAO, 2024) introduced a method that collapses later layers into earlier ones, reducing model size while preserving its structure. In a similar direction, Gromov et al. (GROMOV et al., 2025) focused on analyzing where knowledge is stored: in deeper or shallower layers.

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 (LI et al., 2021; ZHOU et al., 2021). Structured layer pruning is particularly attractive because it shortens the execution path by reducing the number of sequential operations, thereby significantly speeding up latency. 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 work fills this gap. It is worth mentioning that the work by Pons et al. (PONS et al., 2024) constitutes a preliminary effort in this direction. Specifically, the authors introduced a layer-pruning technique. Unfortunately, the number

of layers constrains the maximum computational benefit the method can achieve (other layer-pruning methods suffer from the same limitation). 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 work as it carefully and systematically chooses among the type of structure.

3.3 Pattern Pruning

Pattern pruning (a.k.a semi-structured pruning) bridges the gap between unstructured and structured pruning methods: 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 (MA et al., 2020; ZHOU et al., 2021; LI et al., 2021) (see Figure 5 (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 minimize redundant loads (NIU et al., 2020; HE; XIAO, 2023).

Modern pattern pruning strategies adopt an $N : M$ pruning mask (e.g., 2:4 or 4:8) (SUN et al., 2024; LIANG et al., 2025). Specifically, $N : M$ sparsity forces exactly N out of every M consecutive weights to remain nonzero, leveraging advanced GPU technologies to achieve practical speed-ups (ZHOU et al., 2021).

While $N : M$ sparsity is a prevalent approach, alternative pattern-based methods exist. For example, Ma et al. (MA et al., 2020) 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. Niu et al. (NIU et al., 2020) extended this approach by systematically pruning a fixed number of weights per filter.

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 (LI et al., 2021; ZHAN et al., 2021). Therefore, due to those limitations, our work focus in a multi-structure pruning approach.

3.4 Multi-Structure Pruning

In Multi-Structure Pruning, the main goal is to remove any available structure, regardless of its type, as long as it produces the greatest impact on the architecture. To the best of our knowledge, there is a lack of studies on this field. However, closely related to ours, Muralidharan et al. (MURALIDHARAN et al., 2024) 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.

3.5 *Where are we?* - Overview

Model compression aims to produce efficient neural network architectures while preserving predictive performance. Among the available compression techniques, pruning has emerged as one of the most effective approaches because it removes redundant parameters or architectural components to reduce model complexity. In this dissertation, we categorize pruning methods into four families: unstructured, structured, pattern, and multi-structure pruning.

Unstructured pruning removes individual weights regardless of their location in the network, providing the finest level of pruning granularity. This approach can achieve high sparsity levels, as demonstrated by the Lottery Ticket Hypothesis (FRANKLE; CARBIN, 2019). However, the resulting irregular sparsity often prevents general-purpose hardware from efficiently exploiting the reduced number of parameters, creating a gap between theoretical compression gains and practical inference speedups.

To overcome these hardware limitations, structured pruning removes entire architectural components, such as filters, attention heads, or layers. Filter pruning reduces the width of the network, whereas layer pruning shortens the computational path, leading to lower inference latency. Despite their practical advantages, most existing methods optimize a single type of structure in isolation. As a result, they fail to exploit the complementary contributions and interactions among different structural components through joint optimization.

Pattern pruning combines the advantages of unstructured and structured pruning

by enforcing regular sparsity patterns, such as $N : M$ masks. This strategy achieves high compression rates while remaining compatible with modern hardware accelerators, particularly GPUs that support structured sparse computation. However, its effectiveness depends on hardware support and compiler optimizations, and its performance gains may vary across different network architectures and layer types.

Multi-structure pruning emerges as a promising research direction. Unlike conventional pruning methods that optimize a single architectural component, multi-structure pruning jointly considers multiple structures to reduce model complexity while preserving predictive performance. Despite its potential, research in this area remains limited. Existing approaches either rely on empirical trial-and-error strategies (MURALIDHARAN et al., 2024) or adopt sequential pruning pipelines, such as removing layers before filters, which may overlook the interactions among different structural components (PONS et al., 2024). These limitations motivate the development of more flexible pruning strategies that jointly optimize multiple structures.

To address this gap, we propose a multi-structure pruning approach based on similarity metrics. Rather than relying on heuristics or predefined pruning sequences, the proposed method automatically determines whether to prune filters or layers according to the similarity between architectural components. This strategy enables joint optimization across multiple structures, mitigates the limitations of structure-specific pruning, and improves the balance between computational efficiency and predictive performance. Table 1 gives an overview of the pruning context and answers the question: *Where are we?*

Table 1 – Comparison of Pruning Approaches and the Proposed Multi-Structure Method

Pruning Categories	**Work/Venue**	**Main Characteristics**	**Target**
Unstructured	(HAN et al., 2015) (NeurIPS, 2015)	ℓ_1-norm based weight removal	Weights
	(FRANKLE; CARBIN, 2019) (ICLR, 2019)	Lottery Ticket Hypothesis (Winning Tickets)	Weights
	(FRANTAR et al., 2023) (ICLR, 2023)	Sparse regression solve	Weights
	(SUN et al., 2024) (ICLR, 2024)	Weight magnitude + input activations	Weights
Structured (Filters)	(LI et al., 2017) (ICLR, 2017)	Extension of ℓ_1-norm to filters	Filters
	(GUPTA et al., 2024) (WACV, 2024)	Physics-inspired torque criterion	Filters
	(GAO et al., 2024b) (CVPR, 2024)	Bi-level optimization (Static + Dynamic)	Filters
	(HE et al., 2020) (CVPR,2020)	Adaptive layer-by-layer parameters	Filters
Structured (Layers)	(CHEN; ZHAO, 2019) (TPAMI, 2019)	Block-based layer removal	Layers
	(GANJDANESH; GAO; HUANG, 2024) (CVPR, 2024)	Reinforcement Learning (RL) iterative pruning	Layers
	(FU et al., 2022) (ICML, 2022)	Structural reparameterization and merging	Layers
	(KIM et al., 2024b) (ICML, 2024)	Layer merge with activation removal	Layers
	(PONS et al., 2024) (ICPR, 2024)	Sequential removal (Layers → Filters)	Layers
Pattern-based	(MA et al., 2020) (AAAI, 2020)	Image theory patterns + convex optimization	Local Weight Patterns
Multi-Structure	(MURALIDHARAN et al., 2024) (NeurIPS, 2024)	Alternating width and depth pruning	Filters and Layers
Ours	**Proposed Method**	**Systematic selection via similarity metrics**	**Filters and Layers**

4 Methodology

4.1 Problem Statement

According to recent literature (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b; KIM et al., 2024a), 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 (LI et al., 2022), confirming the potential of random pruning.

4.2 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. To accomplish this, consider a similarity metric that takes the internal representations (a.k.a 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 (KORNBLITH et al., 2019), defined as the normalized Hilbert-Schmidt Independence Criterion (HSIC) by Gretton et al. (GRETTON et al., 2005). Formally, let k and l be two kernels, the empirical estimator of HSIC with m examples for the internal representation extraction is:

$$\text{HSIC}(K, L) = \frac{1}{(m-1)^2} tr(KHLH), \tag{4.1}$$

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 be the internal representation from the parent model $\mathcal{F}$ and $R_{\mathcal{F}'}$ the internal representation from a pruned model $\mathcal{F}'$.

Following Kornblith et al. (KORNBLITH et al., 2019), we compute the CKA in terms of

$$\text{CKA}(R, R_{\mathcal{F}'}) = \frac{\text{HSIC}(R, R_{\mathcal{F}'})}{\sqrt{\text{HSIC}(R, R) \times \text{HSIC}(R_{\mathcal{F}'}, R_{\mathcal{F}'})}}. \tag{4.2}$$

The output from $\text{CKA}(R, R_{\mathcal{F}'}) \in [0, 1]$, where higher values indicate greater similarity between the representations. For simplicity, we slightly abuse notation and apply CKA similarity 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 (i.e., feature maps) of the models.

4.3 Proposed Method

Building upon the previous definitions, 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 (JORDAO et al., 2023), 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 (JORDAO et al., 2023; PONS et al., 2024). To deal with this issue, we apply the KL-divergence criterion by Luo et al. (LUO; WU, 2020). 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 (PONS et al., 2024).

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 6), 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 6 (top)). To guide this decision, we apply the CKA similarity metric from Equation 4.2 and formulate the expression:

$$\mathcal{F} \leftarrow \begin{cases} \mathcal{F}_l^{'} & \text{if } \text{CKA}(\mathcal{F}, \mathcal{F}_l^{'}) \geq \text{CKA}(\mathcal{F}, \mathcal{F}_f^{'}) \\ \mathcal{F}_f^{'} & \text{otherwise.} \end{cases}. \tag{4.3}$$

Due to the iterative nature of our modeling, in Equation 4.3, we slightly abuse notation[1] and assign $\mathcal{F}$ to its pruned version for easy of exposition. Figure 6 illustrates a conceptual

[1] Another way to express this formulation is $\mathcal{F}^k \leftarrow (\cdot)$, where k indicates the kth iteration.

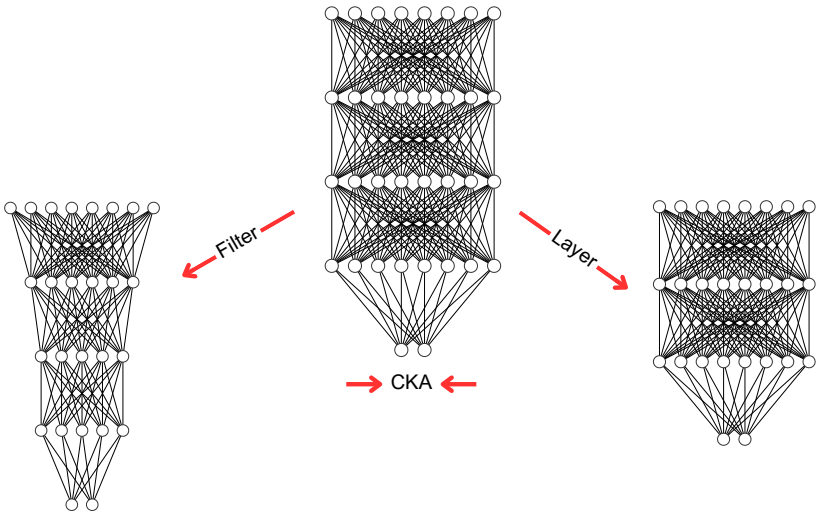


Figure 6 – Proposed method at starting point.

overview of Equation 4.3. Finally, we apply a standard (i.e., supervised) fine-tuning, as this step helps recover predictive ability after pruning (MASON-WILLIAMS; DAHLQVIST, 2024; KIM et al., 2024a; MURALIDHARAN et al., 2024). Importantly, when the similarity of both subnetworks is equal, our method prefers to remove the layer, as this form of pruning promotes additional performance gains (KIM et al., 2024a; PONS et al., 2024). In this sense, Equation 4.3 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 2 summarizes the steps composing our method. From Algorithm 2, we highlight two key observations. (i) Line 4 indicates a branch (i.e., decision) in Figure 1. (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). At the end of each iteration k, we apply standard fine-tuning, see line 9. We refer to this procedure as standard because it follows the same fine-tuning strategy in the state-of-the-art methods, without incorporating additional optimization techniques such as knowledge distillation.

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 2 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 $\mathcal{F}$

for $k \leftarrow 1$ **to** K **do**
  $\mathcal{F}_l^{'} \leftarrow \mathcal{F} \setminus c(\mathcal{F})$ ▷ Remove unimportant layers based on c
  $\mathcal{F}_f^{'} \leftarrow \mathcal{F} \setminus c(\mathcal{F})$ ▷ Remove unimportant filters based on c
  if $\text{CKA}(\mathcal{F}, \mathcal{F}_l^{'}) \geq \text{CKA}(\mathcal{F}, \mathcal{F}_f^{'})$ **then**
    $\mathcal{F} \leftarrow \mathcal{F}_l^{'}$ ▷ $\mathcal{F}$ becomes its layer pruned version
  else
    $\mathcal{F} \leftarrow \mathcal{F}_f^{'}$ ▷ $\mathcal{F}$ becomes its filter pruned version
  end if
  Update $\mathcal{F}$ via standard fine-tuning on $\mathbf{X}$ and $\mathbf{Y}$
end for

5 Experiments

5.1 Experimental Setup

Following the previous literature on pruning (PONS et al., 2024; WU et al., 2024; ALWANI; WANG; MADHAVAN, 2022), we conduct most experiments on CIFAR-10 and ImageNet using different ResNet architectures (HE et al., 2016). These settings are a standard choice to assess the effectiveness of pruning (WU et al., 2024; GANJDANESH; GAO; HUANG, 2024; GAO et al., 2024b; HE; XIAO, 2023; PONS et al., 2024).

Following previous works (MURALIDHARAN et al., 2024; MASON-WILLIAMS; DAHLQVIST, 2024; SUN et al., 2024) before comparing the two pruned subnetworks in Algorithm 2 (line 4), we also perform a 10-epoch fine-tuning. We omit this step from Algorithm 2 to maintain the simplicity and clarity of our approach. For a fair comparison with the parent model, we discount these epochs from our final fine-tuning (line 9).

For modern Transformer-like architectures, such as Large Language Models, we evaluate LLaMA-2 7B (TOUVRON et al., 2023), LLaMA-3 8B (GRATTAFIORI et al., 2024b), and the multimodal vision-language model LLaVA-1.5 7B (LIU et al., 2024). Based on prior work (MA; FANG; WANG, 2023; KIM et al., 2024b; GAO et al., 2024a; ASHKBOOS et al., 2024; OUDERAA et al., 2024), we evaluate these models on the following benchmarks: WinoGrande (SAKAGUCHI et al., 2019), HellaSwag (ZELLERS et al., 2019), ARC-E and ARC-C (CLARK et al., 2018), and PIQA (BISK et al., 2020).

For LLaVA-1.5 (LIU et al., 2024), we use the LMMs-Eval framework (ZHANG et al., 2025) to evaluate performance on four widely used multimodal benchmarks (GUO et al., 2025b; LUO et al., 2025; ZHAO et al., 2024; CHOW et al., 2024): ScienceQA (LU et al., 2022), VizWiz (GURARI et al., 2018), LLaVA-Bench (In-the-Wild) (LIU et al., 2023b), and MM-Vet (YU et al., 2024). We report the official metric defined for each benchmark.

Overall, we keep the experimental setup as simple as possible to highlight the potential and advantages of our work compared to more elaborate setups on pruning such as knowledge distillation (CHEN; ZHAO, 2019; ZHOU; YEN; YI, 2022; MURALIDHARAN et al., 2024). We also conduct experiments on Transformers using tabular data.

To summarize the results, we adopt common approaches and present the delta in accuracy in percentage points (pp) (HE; XIAO, 2023). In this sense, a negative delta indicates a degradation in accuracy, while a positive delta reveals an improvement. Finally, for each division of FLOPs reduction (%), we highlight the best results in bold in terms of delta in accuracy and FLOPs reduction, separately. Our method produces different subnetworks according to the number of iterations (K). Previous works (PONS et al.,

2024; WU et al., 2024; KASHYAP et al., 2025; WANG et al., 2025) show that reporting different pruned models with different compression rates is a common approach for both CNNs and LLMs. Therefore, we follow this convention and report the different output models with the respective pruning rate throughout the iterative process.

5.2 Comparison with State-of-The-Art

We start our analysis by comparing our work with modern and top-performance pruning techniques. Table 2 summarizes the results on standard FLOPs reduction.

According to Table 2, our approach surpasses most existing techniques, regardless of whether it is pruned by layers or filters. In 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 approach only underperform Pons et al. (PONS et al., 2024). 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 structure ignores the fact that there is much more to explore by systematically alternating between them. The results in Table 2 confirm the superiority of making informed pruning decisions. When considering only FLOPs reductions, our approach outperforms all other works, achieving a reduction of 86.57%.

In ResNet110, the previous behavior repeats itself, and considering the compression level, our approach achieves a reduction in FLOPs of 95.82% while exhibiting a negligible delta in precision. To the best of our knowledge, these values represent the highest reduction achieved solely through pruning, and recent surveys confirm this (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b). Finally, Table 3 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 FLOPs reduction.

The key factor behind the aforementioned breakthroughs is the careful selection of which structures to remove during the pruning process. Mason et al. (MASON-WILLIAMS; DAHLQVIST, 2024) argue that knowing which parameters to remove is more important than quantity, reinforcing the argument behind our method, although it explores a structural perspective. In addition, the concept of pruning seeks to maintain generalization while enhancing compression, resulting in a shallower and/or narrower model. 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

Table 2 – Comparison with state-of-the-art methods on CIFAR-10.

Model	Method	Δ Acc.	FLOPs (%)
ResNet56	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(+) 0.08	26.30
	HALP (SHEN et al., 2022b) (NeurIPS, 2022)	(+) 0.03	33.72
	SOKS (LIU et al., 2023) (TNNLS, 2023)	(+) 0.16	35.91
	(PONS et al., 2024) (ICPR, 2024)	**(+) 1.25**	37.52
	Ours	(+) 1.06	**39.10**
	IFSO (CHENG; ZHANG; SHI, 2024a) (TPAMI, 2024)	(+) 0.27	40.20
	GKP-TMI (ZHONG et al., 2022) (ICLR, 2022)	(+) 0.22	43.23
	GCNP (AL., 2022) (IJCAI, 2022)	(+) 0.13	48.31
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.86	48.78
	Tukan et al. (TUKAN; MUALEM; MAALOUF, 2022) (NeurIPS, 2022)	(-) 0.65	50.00
	IFSO (CHENG; ZHANG; SHI, 2024a) (TPAMI, 2024)	(-) 0.03	50.00
	RLAL (GANJDANESH; GAO; HUANG, 2024) (CVPR, 2024)	(+) 0.45	50.00
	UDSP (GAO et al., 2024b) (CVPR, 2024)	(+) 0.66	50.10
	Ours	**(+) 0.95**	**50.36**
	Ours	**(+) 0.91**	53.70
	GNN-RL (YU; MAZAHERI; JANNESARI, 2022) (ICML, 2022)	(+) 0.10	54.00
	ATO (WU et al., 2024) (CVPR, 2024)	(+) 0.24	55.00
	WhiteBox (ZHANG et al., 2023) (TNNLS, 2023)	(+) 0.28	**55.60**
	CLR-RNF (LIN et al., 2023) (TNNLS, 2023)	(+) 0.01	57.30
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.78	60.04
	Ours	**(+) 0.96**	**62.79**
	ATO (WU et al., 2024) (CVPR, 2024)	(-) 0.02	65.30
	Ours	**(+) 0.53**	**69.24**
	DAIS (GUAN et al., 2023) (TNNLS, 2023)	(-) 1.00	70.90
	Ours	**(+) 0.19**	72.67
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.08	75.05
	GCNP (AL., 2022) (IJCAI, 2022)	(-) 0.97	77.22
	Ours	(-) 0.49	**79.88**
	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(-) 2.41	81.50
	Ours	**(-) 1.36**	**86.37**
ResNet110	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(+) 0.38	35.43
	GKP-TMI (ZHONG et al., 2022) (ICLR, 2022)	(+) 0.64	43.31
	(PONS et al., 2024) (ICPR, 2024)	(+) 1.37	44.73
	Ours	**(+) 1.47**	**48.29**
	Ours	**(+) 0.73**	59.95
	DAIS (GUAN et al., 2023) (TNNLS, 2023)	(-) 0.60	60.00
	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	0.00	**61.78**
	(PONS et al., 2024) (ICPR, 2024)	**(+) 0.89**	65.23
	CRL-RNF (LIN et al., 2023) (TNNLS, 2023)	(+) 0.14	66.00
	WhiteBox (ZHANG et al., 2023) (TNNLS, 2023)	(+) 0.62	66.00
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.80	67.10
	Ours	(+) 0.86	**69.90**
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.59	70.83
	Ours	**(+) 0.60**	**73.16**
	Ours	**(+) 0.49**	75.55
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.23	76.42
	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(-) 0.79	**76.92**
	(PONS et al., 2024) (ICPR, 2024)	**(-) 0.41**	87.61
	Ours	(-) 2.91	**95.82**

Table 3 – Comparison with state-of-the-art pruning methods on ImageNet using ResNet50. Δ Acc. considers Top1 accuracy.

Model	Method	Δ Acc.	FLOPs (%)
ResNet50	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(+) 0.16	13.45
	Ours	**(+) 2.09**	**16.98**
	SOSP (NONNENMACHER et al., 2022) (ICLR, 2022)	(+) 0.41	21.00
	GKP-TMI (ZHONG et al., 2022) (ICLR, 2022)	(-) 0.19	22.50
	(PONS et al., 2024) (ICPR, 2024)	(+) 1.11	22.64
	Ours	**(+) 1.61**	**22.64**
	SOSP (NONNENMACHER et al., 2022) (ICLR, 2022)	(+) 0.45	28.00
	(PONS et al., 2024) (ICPR, 2024)	(+) 0.74	28.30
	Ours	**(+) 1.07**	**28.30**
	GKP-TMI (ZHONG et al., 2022) (ICLR, 2022)	(-) 0.62	33.74
	Ours	**(+) 0.36**	33.96
	LPSR (ZHANG; LIU, 2022) (SPL, 2022)	(-) 0.57	37.38
	Ours	(-) 0.74	39.62
	(PONS et al., 2024) (ICPR, 2024)	(-) 0.18	**39.62**
	CLR-RNF (LIN et al., 2023) (TNNLS, 2023)	(-) 1.16	40.39
	DECORE (ALWANI; WANG; MADHAVAN, 2022) (CVPR, 2022)	(-) 1.57	42.30
	HRank (CVPR, 2020)	(-) 1.17	43.77
	SOSP (NONNENMACHER et al., 2022) (ICLR, 2022)	(-) 0.94	45.00
	Ours	(-) 0.98	45.28
	(PONS et al., 2024) (ICPR, 2024)	(-) 0.90	45.28
	WhiteBox (ZHANG et al., 2023) (TNNLS, 2023)	**(-) 0.83**	**45.60**

accuracy.

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 us 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.

5.3 Comparison with State-of-The-Art in Large Language Models

In the context of LLMs, we begin by comparing our method with state-of-the-art pruning techniques. Table 4 summarizes the results across three standard pruning ratios: 20%, 30%, and 40%. The results show that our approach outperforms most existing methods across the evaluated pruning structures. These experiments indicate that our method is effective for LLM architectures and achieves better performance, with only a gradual degradation as the pruning ratio increases.

In the LLaMA-2 7B model, at 30% rate of pruning, our method outperforms the current state-of-the-art. In particular, our multi-structure approach yields a drop of 7.67pp,

Table 4 – Comparison with state-of-the-art methods on LLMs. We highlight the best results in bold. We run each experiment three times and report the mean and standard deviation.

Pruning Ratio	Method	WinoGrande	HellaSwag	ARC-e	ARC-c	PIQA	Avg
0%	LLaMA-2 7B	69.14	75.99	74.58	46.15	79.11	68.99
20%	SliceGPT (ASHKBOOS et al., 2024) (ICLR, 2024)	65.11	59.04	59.76	37.54	69.42	58.18
	AmoebaLLM (FU et al., 2024) (NeurIPS, 2024)	66.90	70.80	70.30	40.20	76.30	64.90
	PruneNet (SENGUPTA; CHAUDHARY; CHAKRABORTY, 2025) (ICLR, 2025)	66.22	68.37	63.93	38.40	74.76	62.34
	SlimLLM (GUO et al., 2025a) (ICML, 2025)	64.88	70.95	**67.17**	38.99	**78.02**	64.00
	ModHiFi-P (KASHYAP et al., 2025) (NeurIPS, 2025)	64.64	62.70	64.73	38.22	72.79	60.62
	CoMe (WANG et al., 2025) (NeurIPS, 2025)	67.25	68.68	64.10	39.59	72.42	62.41
	LINEARPATCH (CHEN et al., 2025) (NeurIPS, 2025)	**67.40**	69.33	64.35	38.23	73.23	62.51
	Ours	$65.96_{\pm 1.39}$	$\mathbf{71.59}_{\pm 0.67}$	$58.91_{\pm 0.09}$	$\mathbf{43.89}_{\pm 0.26}$	$76.28_{\pm 0.52}$	$\mathbf{65.52}_{\pm 0.16}$
30%	SliceGPT (ASHKBOOS et al., 2024) (ICLR, 2024)	61.33	49.62	51.77	31.23	63.55	51.50
	Shortened LLaMA (KIM et al., 2024b) (ICLR, 2024)	61.09	54.97	45.96	34.81	60.99	51.56
	DISP-LLM (GAO et al., 2024a) (NeurIPS, 2024)	63.93	62.87	60.10	37.03	73.72	59.53
	PruneNet (SENGUPTA; CHAUDHARY; CHAKRABORTY, 2025) (ICLR, 2025)	62.90	63.21	53.37	33.70	72.20	57.08
	Yang et al. (YANG et al., 2025b) (ICML, 2025)	60.77	59.18	58.16	34.22	70.95	56.65
	ModHiFi-P (KASHYAP et al., 2025) (NeurIPS, 2025)	59.35	50.61	53.15	32.50	66.59	52.44
	LINEARPATCH (CHEN et al., 2025) (NeurIPS, 2025)	**66.69**	64.52	60.65	34.81	70.29	59.39
	CoMe (WANG et al., 2025) (NeurIPS, 2025)	63.38	65.83	63.51	35.58	**74.05**	60.47
	Ours	$62.25_{\pm 1.34}$	$\mathbf{66.43}_{\pm 0.75}$	$\mathbf{64.81}_{\pm 1.26}$	$\mathbf{39.53}_{\pm 0.86}$	$73.61_{\pm 0.44}$	$\mathbf{61.32}_{\pm 0.56}$
40%	Yang et al. (YANG et al., 2025b) (ICML, 2025)	56.67	49.00	49.83	30.38	66.10	50.39
	Ours	$\mathbf{59.59}_{\pm 0.55}$	$\mathbf{60.07}_{\pm 1.06}$	$\mathbf{58.91}_{\pm 1.39}$	$\mathbf{34.84}_{\pm 0.60}$	$\mathbf{68.81}_{\pm 1.2}$	$\mathbf{56.44}_{\pm 0.79}$
0%	LLaMA-3 8B	72.77	79.19	77.78	52.90	80.85	72.70
20%	LINEARPATCH (CHEN et al., 2025) (NeurIPS, 2025)	70.17	66.74	60.82	**43.17**	72.85	62.75
	CoMe (WANG et al., 2025) (NeurIPS, 2025)	**70.96**	65.52	64.23	40.44	**73.50**	62.93
	Ours	$69.59_{\pm 0.91}$	$\mathbf{71.43}_{\pm 0.45}$	$\mathbf{64.70}_{\pm 2.86}$	43.03 ± 1.82	72.67 ± 1.97	$\mathbf{64.28}_{\pm 1.57}$
30%	CoMe (WANG et al., 2025) (NeurIPS, 2025)	60.62	50.42	50.67	33.70	67.08	52.50
	Yang et al. (YANG et al., 2025b) (ICML, 2025)	60.30	54.40	54.76	33.19	69.10	54.34
	Ours	$\mathbf{66.67} \pm 1.07$	$\mathbf{64.83}_{\pm 0.63}$	$\mathbf{58.68}_{\pm 1.43}$	$\mathbf{39.82}_{\pm 1.00}$	$\mathbf{70.67}_{\pm 0.82}$	$\mathbf{60.13}_{\pm 0.93}$
40%	Yang et al. (YANG et al., 2025b) (ICML, 2025)	56.27	44.74	46.51	27.30	64.20	47.80
	Ours	$\mathbf{62.43}_{\pm 0.57}$	$\mathbf{55.02}_{\pm 1.77}$	$\mathbf{51.92}_{\pm 4.33}$	$\mathbf{34.19}_{\pm 0.72}$	$\mathbf{66.92}_{\pm 0.61}$	$\mathbf{54.10}_{\pm 1.43}$

outperforming the single-structure layer-pruning method by Wang et al. (WANG et al., 2025), where the authors report a drop of 8.52pp. In a higher pruning ratio, at 40%, our method incurs a drop of 12.55pp on LLaMA-2 7B, whereas Yang et al. (YANG et al., 2025b), who prune blocks based on their entropy, report a drop 6.05 higher than ours.

This trend persists for LLaMA-3 8B: at 30% pruning, our method yields a 12.57pp drop, approximately 6 percentage points lower than competing methods at the same pruning ratio.

Overall, the previous results show that our method outperforms state-of-the-art approaches by a substantial margin, demonstrating the effectiveness of our systematic pruning strategy over methods that target a single structure. Importantly, our approach, originally designed for convolutional neural networks, also generalizes to modern architectures, such as large language models. Therefore, we empirically confirm the robustness and scalability of the proposed method and establish a strong foundation for future research on multi-structure pruning.

Table 5 – Results on LLaVA-1.5 7B with different compression rates. We run each experiment three times and report the mean and standard deviation.

Compression	ScienceQA	VizWiz	MM-Vet	LLaVA-Bench	Mean
0%	68.02	53.52	28.44	59.80	52.44
20%	57.63±1.85	47.73±0.83	23.47±1.14	49.97±3.23	44.67±1.17
30%	55.11±1.50	46.55±6.15	21.19±1.91	43.93±3.21	41.70±2.56
40%	43.73±2.67	40.81±8.51	17.26±1.25	36.00±2.17	34.45±1.87

5.4 Effectiveness in Multimodal Architectures

In prior experiments, we evaluate our method on CNNs and Large Language Models across diverse data types, including images and text. In this experiment, we extend our evaluation to Large Vision-Language Models (LVLMs): multimodal architectures that process both text and image inputs. In particular, we consider LLaVA (LIU et al., 2024), a multimodal model that combines a vision encoder for visual feature extraction with a language decoder responsible for text generation. We restrict pruning to the LLM component, since it is responsible for most parameters in the architecture. In this experiment, we prune the LLaVA-1.5 7B model and evaluate the results on ScienceQA (LU et al., 2022), VizWiz (GURARI et al., 2018), MM-Vet (YU et al., 2024), and LLaVA-Bench (In-the-Wild) (LIU et al., 2023b).

Throughout the experiment, we prune at 20%, 30%, and 40% compression rates. Table 5 reports results. On average, we obtain a drop in accuracy of 7.77, 10.74, and 19.99 percentage points, respectively.

To the best of our knowledge, we are one of the few efforts on pruning in multimodal models; therefore, we do not include state-of-the-art comparisons. However, we can compare these results with the LLM experiments because the multimodal architecture includes an LLM component and we leave the image-related module unpruned. By doing so, we observe performance drops close to those of LLaMA-2 7B, suggesting that our pruning strategy exhibits consistent behavior across both LLM and multimodal settings.

In summary, the previous results show that the proposed method effectively accounts for model architectures during pruning. To the best of our knowledge, this is one of the few studies to explore pruning multiple structures at once in multimodal models, and we believe our results establish a baseline for future work.

5.5 Effectiveness in Shallow Architectures

One of the keys to successful pruning lies in the overparameterized nature of networks, particularly in deep models (TAVARES et al., 2025; CHENG; ZHANG; SHI, 2024b). So far, our experiments confirm the effectiveness of our method on this family of

models. In this experiment, we highlight how the proposed method operates well on shallow architectures. For this purpose, we use the shallow versions of ResNet: ResNet32/44.

From Table 6, by applying our method on ResNet32 and ResNet44, we observe the same behavior as in their deeper versions (ResNet56/110). More concretely, we highlight that the proposed method reduces 61.99% and 71.31% of FLOPs respectively from architectures ResNet32/44, without degrading the model accuracy. By exploring higher compression levels, our method reduces FLOPs by 77.78% on ResNet32 and 83.19% on ResNet44, with a minimal accuracy drop of slightly over one percentage point.

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,

Table 6 – Comparison of state-of-the-art methods on CIFAR-10 using ResNet32/44. For each level of FLOPs reduction, we highlight the best results in bold.

Model	Method	Δ Acc.	FLOPs (%)
ResNet32	IFSO (CHENG; ZHANG; SHI, 2024a) (TPAMI, 2024)	(-) 0.28	40.20
	Ours	**(+) 0.80**	40.71
	GKP-TMI (ZHONG et al., 2022) (ICLR, 2022)	(+) 0.22	43.10
	SOKS (LIU et al., 2023) (TNNLS, 2023)	(-) 0.38	46.85
	(PONS et al., 2024)	(+) 0.68	**47.78**
	IFSO (CHENG; ZHANG; SHI, 2024a) (TPAMI, 2024)	(-) 0.51	50.10
	Ours	(+) 0.59	52.92
	DAIS (GUAN et al., 2023) (TNNLS, 2023)	(+) 0.57	53.90
	SOKS (LIU et al., 2023) (TNNLS, 2023)	(-) 0.80	54.58
	(PONS et al., 2024)	(+) 0.05	54.61
	Ours	**(+) 0.71**	**57.40**
	(PONS et al., 2024)	(-) 0.18	61.44
	Ours	**(+) 0.09**	**61.99**
	Ours	(-) 0.53	67.00
	SOSP (NONNENMACHER et al., 2022) (ICLR, 2022)	**(-) 0.24**	67.36
	Ours	(-) 0.81	**69.56**
	Ours	(-) 1.27	74.79
	Ours	**(-) 1.17**	77.78
	Ours	(-) 7.63	**96.15**
ResNet44	DCP-CAC (TNNLS, 2022)	**(-) 0.03**	50.04
	AGMC (ICCV, 2021)	(-) 0.82	**50.00**
	(PONS et al., 2024)	(+) 0.47	53.27
	Ours	**(+) 0.90**	**56.78**
	(PONS et al., 2024)	(+) 0.22	62.95
	Ours	**(+) 0.66**	**71.31**
	(PONS et al., 2024)	(-) 0.29	72.64
	Ours	(-) **0.07**	**74.72**
	Ours	**(-) 0.31**	77.82
	Ours	(-) 1.26	**83.19**

outperforming existing methods. Our approach pushes the boundaries of compression, reducing 87.06% and 83.19% of the FLOPs on ResNet32/44, with only a small drop in accuracy (e.g., 2.83 pp and 1.26 pp).

Importantly, on ResNet32, we achieve a milestone of 96.15% FLOPs reduction, surpassing any other reports from previous work (HE; XIAO, 2023). This behavior aligns with our analysis of deeper models and reinforces the effectiveness of our method on shallow architectures.

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 (HE; XIAO, 2023; CHENG; ZHANG; SHI, 2024b). Additionally, the previous discussion confirms that our method performs effectively on both deep and shallow networks and establishes compatibility with modern architectures, as we shall see in Section 5.6.

5.6 Results on the Transformer Architecture for Tabular Data

This experiment focuses on demonstrating the potential of the proposed method beyond ResNet architectures. As prior studies indicate, pruning techniques extend to modern neural network architectures, such as Transformers (CHENG; ZHANG; SHI, 2024b). 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, as in the work by Sena et al. (SENA et al., 2021).

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 7 shows the results, where the blue circles represent improvements in accuracy. According to the figure, all the subnetworks show improvements in accuracy. We intend to explore this behavior further in future research, particularly in architectures derived from Transformers, such as LLMs.

We also demonstrate that our method, when pruning Transformers remains effective even when the pruning criterion changes. As a concrete example, we prune the model using Kullback-Leibler divergence (KL) and the ℓ_1-norm (L1). Figure 8 shows the outcomes of the experiment and indicates that both criteria perform well.

Overall, the findings of this experiment shows a breakthrough compared to structural pruning on ResNet architectures, and Figure 7 confirms that our method applies effectively to Transformers for tabular data.

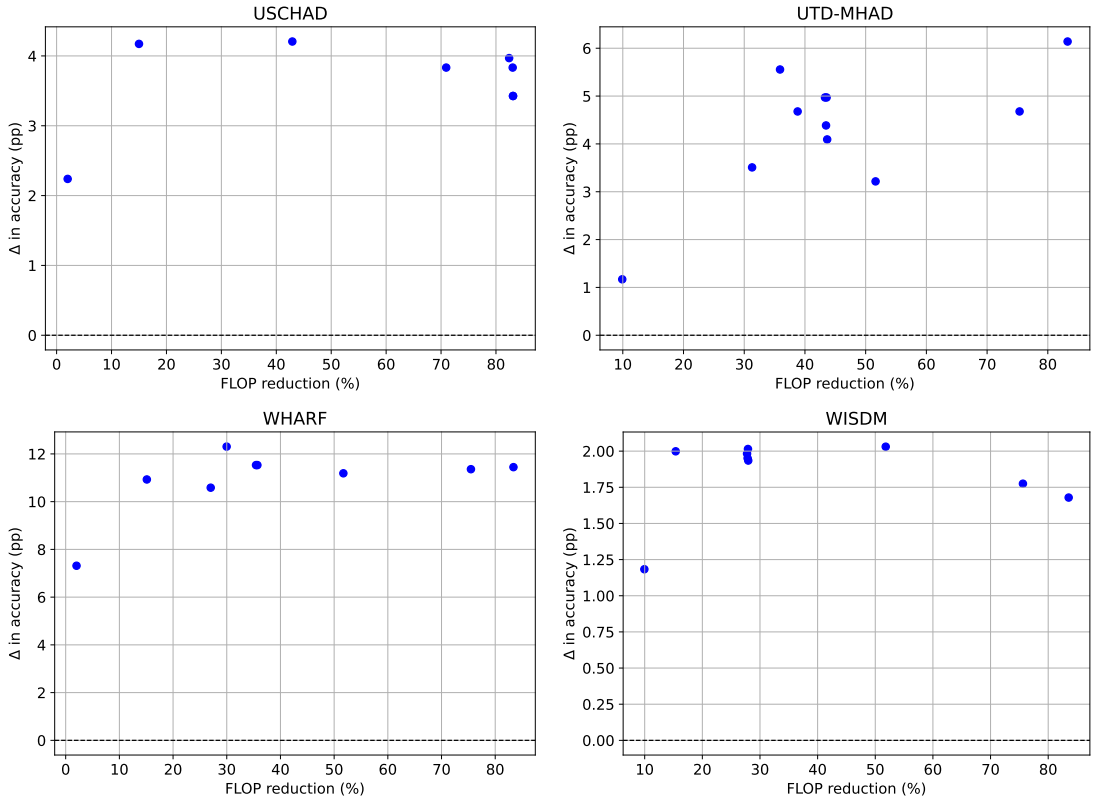


Figure 7 – Performance of our work on Transformer architecture for sensor data from human activity (SENA et al., 2021). Each circle represents a pruned model. A blue circle denotes a subnetwork with an improvement in accuracy compared to the unpruned model.

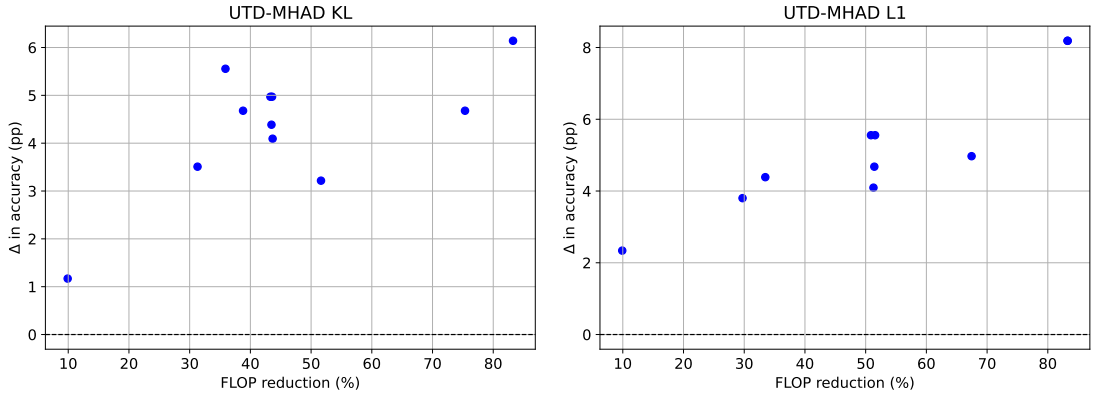


Figure 8 – Performance on Transformer architecture with different pruning criteria c.

5.7 Robustness to Pruning Criteria

Once we confirmed the effectiveness of our pruning method across architectures and against state-of-the-art approaches (PONS et al., 2024; SENGUPTA; CHAUDHARY; CHAKRABORTY, 2025; KASHYAP et al., 2025; ALWANI; WANG; MADHAVAN, 2022; TUKAN; MUALEM; MAALOUF, 2022; NONNENMACHER et al., 2022), we now turn our attention to demonstrate the robustness of our method by illustrating its flexibility with respect to popular pruning criteria c. For this purpose, we prune ResNet32 using two

Table 7 – Results on CIFAR-10 using ResNet32 with different criteria c.

Model	Method	Δ Acc.	FLOPs (%)
ResNet32	ℓ_1-norm	+0.22	54.60
	KL	+0.09	**61.99**

criteria with different natures: the ℓ_1-norm (L1), and compare it with the Kullback-Leibler (KL). The L1 plays a central role in pruning, particularly in approaches related to the Lottery Ticket Hypothesis, as well as in filter (structured) and weight (unstructured) pruning (FRANKLE; CARBIN, 2019). In contrast, the Kullback-Leibler (KL) divergence is frequently applied to the softmax output distributions of the parent model and the subnetwork, which makes it agnostic to whether the pruning strategy follows a structured or unstructured approach. Table 7 shows the results, where our method maintains a positive delta in accuracy. From these results, we observe a slight difference between the to type of pruning criteria, and theirs impacts in how further it is possible to go without degradation. Therefore, we confirm that our method successfully works with other criteria.

5.8 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 this purpose, we prune the architectures ResNet32/44/56 by randomly selecting whether to remove filters or layers (namely, Random Walk). In other words, we replace line 4 in Algorithm 2 with a coin flip. Importantly, we compare our method with Random Walk, varying only the selection mechanism, while all other settings (i.e., initialization, training/fine-tuning epochs, data augmentation and so on) remain identical. From these settings, we force pruning until the pruned model exhibits no degradation (positive delta) compared to the original, unpruned model. Due to the stochastic nature of the Random Walk, we ran it three times and averaged the results.

Table 8 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,

Table 8 – Comparison with random selection baseline (Random Walk) on CIFAR-10 using different ResNets. Although Random Walk highlights positive accuracy, it stops pruning earlier.

Model	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**

as the pruning proceed further without compromising accuracy. In our initial analysis, we considered using some heuristics such as A* and branch-and-bound. However, their time complexities are computationally prohibitive due to the fine-tuning steps. Therefore, we do not extend experiments using these and other classical optimization mechanisms.

5.9 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 2), 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 and ResNet110. 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, the above 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.*

5.10 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 (MASON-WILLIAMS; DAHLQVIST, 2024; SUN et al., 2024; MURALIDHARAN et al., 2024), 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, Mason et al. (MASON-WILLIAMS; DAHLQVIST, 2024) 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 (MASON-WILLIAMS; DAHLQVIST, 2024; SUN et al., 2024), 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. We note that after fine-tuning the method favors no structure; therefore, this step adheres to the principle of recovering the subnetwork and determining which one is more similar to its parent.

It is worth mentioning that as we follow this recovery strategy, there is no effect on our fine-tuning process in Algorithm 2 (line 9), as we discount the amount of 10-epoch from the remaining fine-tuning epochs.

5.11 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 (WILLIAMS et al., 2021), Gaussian Stochastic and Wasserstein (DUONG et al., 2023) metrics. According to previous works (DUONG et al., 2023; WILLIAMS et al., 2021), 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.

Table 9 – Comparison with other similarity metrics on CIFAR-10. These results reinforce that our method is a similarity metric-agnostic strategy.

Model	Method	Δ Acc.	FLOPs (%)
ResNet32	Linear	+0.24	56.18
	Gaussian Stochastic	+0.71	47.78
	Wasserstein	+0.32	52.50
	Ours (CKA)	+0.09	61.99

Table 9 summarizes the results alongside with the result on ResNet32 with CKA. 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.

5.12 Robustness to Adversarial Samples

As deep models assume greater significance in critical scenarios such as medical diagnosis or financial systems, robustness to out-of-distribution (OOD) and adversarial samples becomes equally important as the generalization itself (JORDAO et al., 2023; BAIR et al., 2024). The main reason is that in real-world applications, reliability when faced with this type of input is crucial for decision-making tasks, preventing undesirable outcomes and potential catastrophes. In this context, previous works studied the impact of pruning on adversarial robustness and OOD generalization, supporting that pruning mechanisms enhance these metrics (JORDAO et al., 2023; BAIR et al., 2024).

In this experiment, we aim to assess the impact of multiple structure removal promoted by our technique on adversarial attacks and OOD generalization by considering widely adopted benchmarks. To achieve this, we employ the CIFAR-C (HENDRYCKS; DIETTERICH, 2019) and CIFAR-10.2 (LU et al., 2020) datasets and FGSM attack to test our pruned models. Specifically, we achieve improvements in robustness even at high compression rates, for example, 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 work 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. These results confirm the effectiveness of our approach and highlight the potential of pruning as a mechanism capable of improving robustness. Importantly, these results confirm the applicability of our pruned models in critical scenarios.

5.13 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 (LACOSTE et al., 2019; FAIZ et al., 2024; MORRISON et al., 2025). 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 (LACOSTE et al., 2019). In summary, we provide solid results toward GreenAI, helping to reduce CO_2 emissions and making deep models more financially accessible.

5.14 Deployment on Low-Cost Devices

One objective of pruning a neural network is to obtain a smaller subnetwork that preserves generalization. Another important goal is to deploy these subnetworks on low-cost devices. Within this context, we now turn our attention to highlights the improvements in latency when uploading compressed models to resource-constrained hardware. To conduct this experiment, we deploy the pruned models on an ESP32-CAM with 520 KB of RAM and an Xtensa dual-core 32-bit LX6 processor and a 2MP camera (OV2640). Figure 9 shows the inference speedup on ResNet32, ResNet44, ResNet56, and ResNet110 as a function of FLOP reduction.

In our pruning process, each iteration produces the best subnetwork found at that moment. In this experiment, we also attempt to deploy each of these subnetworks to the ESP32-CAM. However, some of them are too large to run on the device. Therefore, Figure 9 shows the latency of the models that successfully executed inference on this hardware.

In general, this experiment demonstrates the importance of our method, and a key breakthrough is that most of the architectures require more than 70% compression to become feasible for deployment. Therefore, as we achieve high levels of compression, we manage to deploy large CNNs to low-cost devices, which would otherwise be infeasible without pruning.

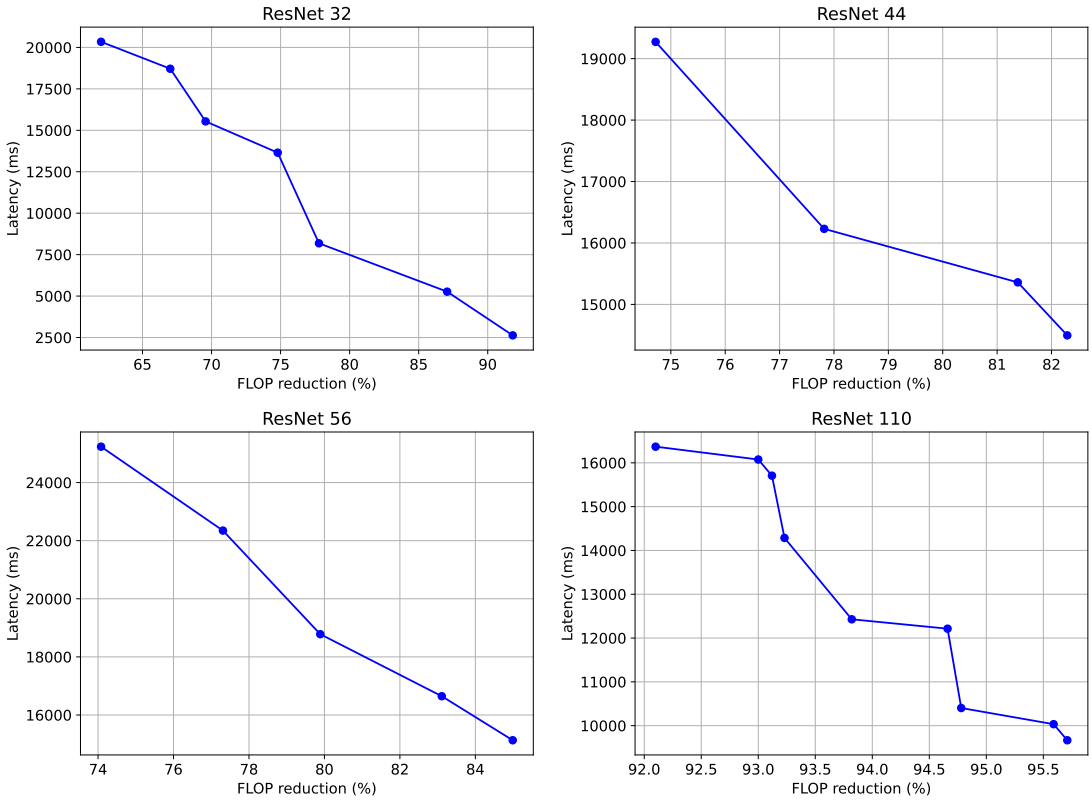


Figure 9 – Speed-up of the pruned models of ResNet 32/44/56/110.

6 Conclusions

Modern pruning approaches exhibit promising results and distinct computational benefits in removing neurons (filters, heads) 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, in a broad of different architectures. 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. On convolutional neural networks, we achieve milestones in FLOP reduction such as a 95.82% FLOP reduction on ResNet110. Such achievements come from the fact that our method leverages the best of both layer and neuron pruning, while existing methods are confined to a single structure. Also, as a consequence of these achievements, these results enable deployment of highly compressed ResNets on low-cost devices, which would otherwise be infeasible without the proposed method.

In large language models, we jointly prune multiple structural components, a setting that, to the best of our knowledge, has not been explored in prior work. At the commonly used 30% pruning rate, our method outperforms all compared methods. We observe a drop of 7.67 percentage points in accuracy on LLaMA-2 7B and 12.57 on LLaMA-3 8B, which is 5.79 percentage points lower than the state of the art. A key milestone of this extension is the evaluation on a multimodal model, LLaVA-1.5 7B, where we successfully prune at 20%, 30%, and 40% compression ratios.

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.

6.1 Future Work

Despite our contributions, several directions remain open for future investigation. First, future work should extend our pruning strategy by exploring more sophisticated decision mechanisms for selecting the structure to remove at each pruning step. In the current formulation, the decision relies on the similarity between the subnetworks and their parent network. Future studies should investigate adaptive strategies that combine similarity with additional information, such as computational cost, parameter distribution, sensitivity, or the expected impact on downstream performance. Such strategies would enable more informed selection of the structures to prune and potentially improve the trade-off between compression and accuracy.

Another promising direction concerns the investigation of similarity metrics for guiding the pruning process. Although our results demonstrate that the proposed method is similarity metric-agnostic, different metrics may capture complementary aspects of internal representations. A systematic analysis of these properties could provide a better understanding of how representation similarity relates to pruning decisions and model performance. This analysis could also investigate whether dynamically selecting or combining similarity metrics during the pruning process provides additional benefits.

The extension of the proposed method to larger and more diverse foundation models also remains an important direction. Our experiments demonstrate the applicability of the approach to autoregressive language models and to a multimodal architecture through LLaVA-1.5 7B. Future work could investigate larger language and multimodal models, as well as architectures with substantially different structural configurations. In particular, multimodal models provide an opportunity to study whether jointly pruning components from different modalities can preserve the representations required for both language and visual reasoning.

Furthermore, the robustness observed in the pruned models motivates a deeper investigation into the relationship between pruning and adversarial robustness. Future experiments could evaluate a broader range of adversarial attacks, threat models, and pruning configurations to determine whether the preservation of internal representations contributes systematically to robustness. Understanding this relationship could lead to pruning strategies that jointly optimize computational efficiency, predictive performance, and robustness.

Finally, future work should evaluate the proposed method beyond theoretical computational savings and benchmark accuracy. Measuring inference latency, memory consumption, energy usage, and carbon emissions on representative low-cost and edge hardware would provide a more comprehensive assessment of the practical benefits of the method. Such experiments could determine whether the reductions in FLOPs and training costs consistently translate into real-world efficiency gains, strengthening the connection between model pruning and GreenAI.

Bibliography

AL., D. J. et. On the channel pruning using graph convolution network for convolutional neural network acceleration. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. [S.l.: s.n.], 2022. Cited in page 43.

ALAYRAC, J.-B. et al. Flamingo: a visual language model for few-shot learning. *ArXiv*, 2022. Cited in page 28.

ALWANI, M.; WANG, Y.; MADHAVAN, V. DECORE: deep compression with reinforcement learning. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2022. Cited 4 times in pages 41, 43, 44, and 49.

ASHKBOOS, S. et al. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2024. Cited 2 times in pages 41 and 45.

BAI, Y. et al. Constitutional ai: Harmlessness from ai feedback. *ArXiv*, 2022. Cited in page 27.

BAIR, A. et al. Adaptive sharpness-aware pruning for robust sparse networks. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2024. Cited in page 53.

BENDER, E. M. et al. On the dangers of stochastic parrots: Can language models be too big? In: *Conference on Fairness, Accountability, and Transparency (FAT)*. [S.l.: s.n.], 2021. Cited in page 27.

BENGIO, Y. et al. International ai safety report. 2025. Cited 2 times in pages 13 and 19.

BISK, Y. et al. PIQA: Reasoning about Physical Commonsense in Natural Language. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. [S.l.: s.n.], 2020. Cited in page 41.

BOMMASANI, R. et al. On the opportunities and risks of foundation models. *ArXiv*, 2021. Cited in page 13.

BROWN, T. B. et al. Language models are few-shot learners. 2020. Cited 2 times in pages 14 and 26.

CHEN, S.; ZHAO, Q. Shallowing deep networks: Layer-wise pruning based on feature representations. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019. Cited 3 times in pages 33, 36, and 41.

CHEN, X. et al. A Simple Linear Patch Revives Layer-Pruned Large Language Models. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2025. Cited in page 45.

CHENG, H.; ZHANG, M.; SHI, J. Q. Influence function based second-order channel pruning: Evaluating true loss changes for pruning is possible without retraining. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. Cited 2 times in pages 43 and 47.

CHENG, H.; ZHANG, M.; SHI, J. Q. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. Cited 11 times in pages 13, 14, 15, 24, 30, 31, 32, 37, 42, 46, and 48.

CHOW, W. et al. Unified Generative and Discriminative Training for Multi-modal Large Language Models. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2024. Cited in page 41.

CLARK, P. et al. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *ArXiv*, 2018. Cited in page 41.

DEEPSEEK-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, 2025. Cited in page 13.

DEVLIN, J. et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: *North Chapter Association for Linguistics (NAACL)*. [S.l.: s.n.], 2019. Cited in page 26.

DIXON, L. et al. Watch what i say: Stylistic contamination in named entity recognition systems. In: *Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2018. Cited in page 27.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021. Cited in page 27.

DROR, A. B. et al. Layer folding: Neural network depth reduction using activation linearization. In: *British Machine Vision Conference (BMVC)*. [S.l.: s.n.], 2022. Cited in page 33.

DUONG, L. R. et al. Representational dissimilarity metric spaces for stochastic neural networks. *International Conference on Learning Representations (ICLR)*, 2023. Cited 2 times in pages 24 and 52.

ELSEN, E. et al. Fast sparse convnets. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Cited 2 times in pages 31 and 32.

FAIZ, A. et al. LLMCarbon: Modeling the end-to-end carbon footprint of large language models. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2024. Cited 4 times in pages 13, 19, 24, and 54.

FRANKLE, J.; CARBIN, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2019. Cited 4 times in pages 31, 35, 36, and 50.

FRANTAR, E.; ALISTARH, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2023. Cited in page 27.

FRANTAR, E. et al. Optq: Accurate quantization for generative pre-trained transformers. 2023. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2023. Cited 3 times in pages 13, 31, and 36.

FU, Y. et al. Depthshrinker: A new compression paradigm towards boosting real-hardware efficiency of compact neural networks. In: *International Conference on Machine Learning (International Conference on Machine Learning (ICML))*. [S.l.: s.n.], 2022. Cited 2 times in pages 33 and 36.

FU, Y. et al. AmoebaLLM: Constructing Any-Shape Large Language Models for Efficient and Instant Deployment. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2024. Cited in page 45.

GANJDANESH, A.; GAO, S.; HUANG, H. Jointly training and pruning cnns via learnable agent guidance and alignment. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. Cited 5 times in pages 16, 33, 36, 41, and 43.

GAO, S. et al. DISP-LLM: Dimension-independent structural pruning for large language models. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2024. Cited 2 times in pages 41 and 45.

GAO, S. et al. Bilevelpruning: Unified dynamic and static channel pruning for convolutional neural networks. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. Cited 4 times in pages 32, 36, 41, and 43.

GEIRHOS, R. et al. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2019. Cited in page 23.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Cited in page 23.

GRATTAFIORI, A. et al. The llama 3 herd of models. *ArXiv*, 2024. Cited in page 19.

GRATTAFIORI, A. et al. The llama 3 herd of models. *ArXiv*, 2024. Cited 3 times in pages 13, 27, and 41.

GRETTON, A. et al. Measuring statistical dependence with hilbert-schmidt norms. In: *Algorithmic Learning Theory (ALT)*. [S.l.: s.n.], 2005. Cited in page 37.

GROMOV, A. et al. The Unreasonable Ineffectiveness of the Deeper Layers. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2025. Cited in page 33.

GUAN, Y. et al. DAIS: automatic channel pruning via differentiable annealing indicator search. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023. Cited 2 times in pages 43 and 47.

GUO, J. et al. SlimLLM: Accurate Structured Pruning for Large Language Models. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2025. Cited in page 45.

GUO, Y. et al. Dynamic Mixture of Experts: An Auto-Tuning Approach for Efficient Transformer Models. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2025. Cited in page 41.

GUPTA, A. et al. Torque based structured pruning for deep neural network. In: *Winter Conference on Applications of Computer Vision (WACV)*. [S.l.: s.n.], 2024. Cited 2 times in pages 32 and 36.

GURARI, D. et al. VizWiz Grand Challenge: Answering Visual Questions from Blind People. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018. Cited 2 times in pages 41 and 46.

HAN, S. et al. Learning both weights and connections for efficient neural network. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2015. Cited 3 times in pages 31, 32, and 36.

HASSIBI, B.; STORK, D. G.; WOLFF, G. J. Optimal brain surgeon and general network pruning. In: *International Conference on Neural Networks (ICNN)*. [S.l.: s.n.], 1993. Cited in page 31.

HE, K. et al. Deep residual learning for image recognition. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Cited 2 times in pages 22 and 41.

HE, Y. et al. Learning filter pruning criteria for deep convolutional neural networks acceleration. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Cited 2 times in pages 32 and 36.

HE, Y.; XIAO, L. Structured pruning for deep convolutional neural networks: A survey. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023. Cited 10 times in pages 14, 15, 30, 31, 32, 34, 37, 41, 42, and 48.

HENDRYCKS, D.; DIETTERICH, T. G. Benchmarking neural network robustness to common corruptions and perturbations. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2019. Cited in page 53.

HOFFMANN, J. et al. Training compute-optimal large language models. *ArXiv*, 2022. Cited 2 times in pages 14 and 26.

JI, Z. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv*, 2023. Cited in page 26.

JORDAO, A. et al. When layers play the lottery, all tickets win at initialization. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2023. Cited 2 times in pages 38 and 53.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. ed. [s.n.], 2026. Online manuscript released August 19, 2026. Available on: <https://web.stanford.edu/~jurafsky/slp3/>. Cited in page 25.

KAPLAN, J. et al. Scaling laws for neural language models. *ArXiv*, 2020. Cited in page 26.

KASHYAP, D. et al. ModHiFi: Identifying High Fidelity predictive components for Model Modification. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2025. Cited 3 times in pages 42, 45, and 49.

KIM, B.-K. et al. Shortened LLaMA: A simple depth pruning for large language models. In: *Workshop on Understanding of Foundation Models at International Conference on Learning Representations (ME-FoMo ICLR)*. [S.l.: s.n.], 2024. Cited 2 times in pages 37 and 39.

KIM, J. et al. Layermerge: Neural network depth compression through layer pruning and merging. *International Conference on Machine Learning (ICML)*, 2024. Cited 5 times in pages 14, 33, 36, 41, and 45.

KLABUNDE, M. et al. Resi: A comprehensive benchmark for representational similarity measures. *International Conference on Learning Representations (ICLR)*, 2025. Cited in page 24.

KORNBLITH, S. et al. Similarity of neural network representations revisited. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2019. Cited 3 times in pages 16, 37, and 38.

LACOSTE, A. et al. Quantifying the carbon emissions of machine learning. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2019. Cited 2 times in pages 19 and 54.

LECUN, Y.; DENKER, J.; SOLLA, S. Optimal brain damage. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 1989. Cited in page 31.

LEWIS, M. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. Cited in page 27.

LI, H. et al. Pruning filters for efficient convnets. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2017. Cited 2 times in pages 32 and 36.

LI, J. et al. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2023. Cited in page 28.

LI, Y. et al. Revisiting random channel pruning for neural network compression. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2022. Cited in page 37.

LI, Z. et al. NPAS: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2021. Cited 2 times in pages 33 and 34.

LIANG, Y. et al. Efficientllava:generalizable auto-pruning for large vision-language models. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2025. Cited in page 34.

LIN, M. et al. Pruning networks with cross-layer ranking & k-reciprocal nearest filters. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023. Cited 2 times in pages 43 and 44.

LIN, M. et al. Hrank: Filter pruning using high-rank feature map. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Cited in page 32.

LIU, H. et al. Improved Baselines with Visual Instruction Tuning. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. Cited 3 times in pages 27, 41, and 46.

LIU, H. et al. Visual instruction tuning. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2023. Cited 3 times in pages 14, 27, and 28.

LIU, H. et al. Visual Instruction Tuning. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2023. Cited 2 times in pages 41 and 46.

LIU, L. et al. Group fisher pruning for practical network compression. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2021. Cited in page 32.

LIU, S. et al. A survey on model compression and acceleration for large language models. *ArXiv*, 2023. Cited 3 times in pages 26, 43, and 47.

LU, P. et al. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2022. Cited 2 times in pages 41 and 46.

LU, S. et al. Harder or different? a closer look at distribution shift in dataset reproduction. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2020. Cited in page 53.

LUO, G. et al. Feast Your Eyes: Mixture-of-Resolution Adaptation for Multimodal Large Language Models. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2025. Cited in page 41.

LUO, J.-H.; WU, J. Neural network pruning with residual-connections and limited-data. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Cited in page 38.

LUO, W. et al. Benchmarking and dissecting the nvidia hopper GPU architecture. In: *International Parallel and Distributed Processing Symposium (IPDPS)*. [S.l.: s.n.], 2024. Cited in page 31.

LUO, X. et al. Icp: Immediate compensation pruning for mid-to-high sparsity. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2025. Cited 2 times in pages 24 and 33.

MA, X.; FANG, G.; WANG, X. LLM-Pruner: On the Structural Pruning of Large Language Models. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2023. Cited in page 41.

MA, X. et al. PCONV: the missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. [S.l.: s.n.], 2020. Cited 2 times in pages 34 and 36.

MASLEJ, N. et al. Artificial intelligence index report. 2024. Cited in page 13.

MASLEJ, N. et al. Artificial intelligence index report. 2025. Cited in page 13.

MASON-WILLIAMS, G.; DAHLQVIST, F. What makes a good prune? maximal unstructured pruning for maximal cosine similarity. 2024. Cited 5 times in pages 16, 39, 41, 42, and 52.

MICHEL, P.; LEVY, O.; NEUBIG, G. Are sixteen heads really better than one? *Neural Information Processing Systems (NeurIPS)*, 2019. Cited in page 26.

MITTAL, D. et al. Recovering from random pruning: On the plasticity of deep convolutional neural networks. In: *Winter Conference on Applications of Computer Vision (WACV)*. [S.l.: s.n.], 2018. Cited in page 21.

MORRISON, J. et al. Holistically evaluating the environmental impact of creating language models. 2025. Cited 3 times in pages 13, 19, and 54.

MURALIDHARAN, S. et al. Compact language models via pruning and knowledge distillation. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2024. Cited 7 times in pages 16, 30, 35, 36, 39, 41, and 52.

NIU, W. et al. Patdnn: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning. In: *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. [S.l.: s.n.], 2020. Cited 2 times in pages 32 and 34.

NONNENMACHER, M. et al. SOSP: efficiently capturing global correlations by second-order structured pruning. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2022. Cited 3 times in pages 44, 47, and 49.

OpenAI. *GPT-4 Technical Report*. 2023. Cited 2 times in pages 14 and 26.

OUDERAA, T. F. A. van der et al. The LLM Surgeon. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2024. Cited in page 41.

PATTERSON, D. et al. Carbon emissions and large neural language model training. *Empirical Methods in Natural Language Processing (EMNLP)*, 2021. Cited 2 times in pages 14 and 27.

PONS, I. et al. Effective layer pruning through similarity metric perspective. In: *International Conference on Pattern Recognition (ICPR)*. [S.l.: s.n.], 2024. Cited 13 times in pages 14, 16, 30, 33, 36, 38, 39, 41, 42, 43, 44, 47, and 49.

RADFORD, A. et al. Learning transferable visual models from natural language supervision. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2021. Cited in page 28.

RAFFEL, C. et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 2020. Cited in page 26.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4th. ed. [S.l.]: Pearson, 2021. Cited in page 23.

SAKAGUCHI, K. et al. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *ArXiv*, 2019. Cited in page 41.

SENA, J. et al. Human activity recognition based on smartphone and wearable sensors using multiscale dcnn ensemble. *Neurocomputing*, 2021. Cited 2 times in pages 48 and 49.

SENGUPTA, A.; CHAUDHARY, S.; CHAKRABORTY, T. You Only Prune Once: Designing Calibration-Free Model Compression With Policy Learning. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2025. Cited 2 times in pages 45 and 49.

SHEN, M. et al. When to prune? A policy towards early structural pruning. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2022. Cited in page 31.

SHEN, M. et al. Structural pruning via latency-saliency knapsack. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2022. Cited 2 times in pages 16 and 43.

STRUBELL, E.; GANESH, A.; MCCALLUM, A. Energy and policy considerations for deep learning in nlp. In: *Association for Computational Linguistics (ACL)*. [S.l.: s.n.], 2019. Cited 2 times in pages 14 and 27.

SUN, C. et al. Distillation-based variable budget compression for large language models. In: *Conference on Data Science (CDD)*. [S.l.: s.n.], 2020. Cited in page 26.

SUN, M. et al. A simple and effective pruning approach for large language models. *International Conference on Learning Representations (ICLR)*, 2024. Cited 6 times in pages 30, 31, 34, 36, 41, and 52.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. Cited in page 21.

TAVARES, C. et al. Tiny titans: Efficient large vision, language and multimodal models through pruning. In: *Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.: s.n.], 2025. Cited 2 times in pages 14 and 46.

TOUVRON, H. et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv*, 2023. Cited 2 times in pages 27 and 41.

TUKAN, M.; MUALEM, L.; MAALOUF, A. Pruning neural networks via coresets and convex geometry: Towards no assumptions. 2022. Cited 2 times in pages 43 and 49.

VASWANI, A. et al. Attention is all you need. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2017. Cited 2 times in pages 25 and 26.

VEIT, A.; WILBER, M. J.; BELONGIE, S. J. Residual networks behave like ensembles of relatively shallow networks. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2016. Cited in page 33.

WANG, F. et al. Layer as Puzzle Pieces: Compressing Large Language Models through Layer Concatenation. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2025. Cited 2 times in pages 42 and 45.

WANG, H. et al. Recent advances on neural network pruning at initialization. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. [S.l.: s.n.], 2022. Cited in page 31.

WILLIAMS, A. H. et al. Generalized shape metrics on neural representations. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2021. Cited 2 times in pages 24 and 52.

WU, X. et al. Auto-train-once: Controller network guided automatic network pruning from scratch. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. Cited 4 times in pages 33, 41, 42, and 43.

XIONG, S. et al. Learning feature relationships in cnn model via relational embedding convolution layer. *Neural Networks (NN)*, Elsevier, 2024. Cited in page 23.

XU, C.; MCAULEY, J. J. A survey on model compression and acceleration for pretrained language models. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. [S.l.: s.n.], 2023. Cited in page 30.

YANG, A. et al. Qwen3 technical report. *ArXiv*, 2025. Cited in page 13.

YANG, M. et al. Let LLM Tell What to Prune and How Much to Prune. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2025. Cited in page 45.

YANG, Y.; CAO, Z.; ZHAO, H. Laco: Large language model pruning via layer collapse. *Empirical Methods in Natural Language Processing (EMNLP)*, 2024. Cited 2 times in pages 30 and 33.

YU, S.; MAZAHERI, A.; JANNESARI, A. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2022. Cited in page 43.

YU, W. et al. MM-vet: Evaluating large multimodal models for integrated capabilities. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2024. Cited 2 times in pages 41 and 46.

ZELLERS, R. et al. HellaSwag: Can a Machine Really Finish Your Sentence? In: *Association for Computational Linguistics (ACL)*. [S.l.: s.n.], 2019. Cited in page 41.

ZHAN, Z. et al. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2021. Cited in page 34.

ZHANG, C.; BENGIO, S.; SINGER, Y. Are all layers created equal? *Journal of Machine Learning Research (JMLR)*, 2022. Cited in page 33.

ZHANG, K. et al. LMMs-eval: Reality check on the evaluation of large multimodal models. In: *Association for Computational Linguistics (ACL)*. [S.l.: s.n.], 2025. Cited in page 41.

ZHANG, K.; LIU, G. Layer pruning for obtaining shallower resnets. *Signal Processing Letters (SPL)*, 2022. Cited in page 44.

ZHANG, Y. et al. Carrying out CNN channel pruning in a white box. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023. Cited 2 times in pages 43 and 44.

ZHAO, H. H. et al. LOVA3: Learning to visual question answering, asking and assessment. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2024. Cited in page 41.

ZHAO, W. X. et al. A survey of large language models. *Springer*, 2026. Cited in page 26.

ZHONG, S. et al. Revisit kernel pruning with lottery regulated grouped convolutions. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2022. Cited 3 times in pages 43, 44, and 47.

ZHOU, A. et al. Learning N: M fine-grained structured sparse neural networks from scratch. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2021. Cited 3 times in pages 32, 33, and 34.

ZHOU, Y.; YEN, G. G.; YI, Z. Evolutionary shallowing deep neural networks at block levels. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022. Cited 2 times in pages 33 and 41.