IDAP++: ADVANCING DIVERGENCE-BASED PRUNING VIA FILTER-LEVEL AND LAYER-LEVEL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents a novel approach to neural network compression that addresses redundancy at both the filter and architectural levels through a unified framework grounded in information flow analysis. Building on the concept of tensor flow divergence, which quantifies how information is transformed across network layers, we develop a two-stage optimization process. The first stage employs iterative divergence-aware pruning to identify and remove redundant filters while preserving critical information pathways. The second stage extends this principle to higher-level architecture optimization by analyzing layer-wise contributions to information propagation and selectively eliminating entire layers that demonstrate minimal impact on network performance. The proposed method naturally adapts to diverse architectures, including convolutional networks, transformers, and hybrid designs, providing a consistent metric for comparing the structural importance across different layer types. Experimental validation across multiple modern architectures and datasets reveals that this combined approach achieves substantial model compression while maintaining competitive accuracy. The presented approach achieves parameter reduction results that are globally comparable to those of state-of-the-art solutions and outperforms them across a wide range of modern neural network architectures, from convolutional models to transformers. The results demonstrate how flow divergence serves as an effective guiding principle for both filter-level and layer-level optimization, offering practical benefits for deployment in resource-constrained environments.

1 INTRODUCTION

Modern artificial intelligence (AI) systems are rapidly transforming industries and high-tech products (Jumper et al., 2021; Brown et al., 2020; McKinney et al., 2020; Merchant et al., 2023; Team et al., 2023; Wong et al., 2023). Today, AI powers mobile devices (Liu et al., 2024b; Ignatov et al., 2023), autonomous vehicles (Chen et al., 2024; Kim et al., 2021), healthcare (Cameron et al., 2022; Zarghami, 2024), finance (Iacovides et al., 2024; Rodriguez-Caballero & Villanueva-Domínguez, 2022), industry (Shiue et al., 2018; Jiang et al., 2019), and scientific research (Miret et al., 2024; Wang, 2025). Most of these achievements rely on deep neural networks (DNNs) (Tan & Le, 2019a; Tripp et al., 2024), which over the past decade have revolutionized computer vision (Ravi et al., 2024; Oquab et al., 2024; Zhang et al., 2025), natural language processing (OpenAI et al., 2023; Jiang et al., 2024; Team et al., 2024), generative models (Liu et al., 2024a; Yang et al., 2023; Shi et al., 2023), and control systems (Salzmann et al., 2023; Mu et al., 2022; Ullah et al., 2024). Prominent examples include GPT-4 (Peng et al., 2023), Gemini (Team et al., 2025), medical diagnostic CNNs (Desai, 2024), and image generation models such as DALL·E (Marcus et al., 2022) and Stable Diffusion (Ho et al., 2020; Dhariwal & Nichol, 2021; Ramesh et al., 2022). These advances have enabled unprecedented accuracy and adaptability.

Yet such progress has come with an exponential growth in model scale (Bernstein et al., 2021). State-of-the-art architectures contain hundreds of millions or even billions of parameters, demanding vast computational clusters (Lee et al., 2023; Grattafiori et al., 2024; Kindratenko et al., 2010). The costs include not only training time and energy but also deployment expenses (Baresi & Quattrocchi,

2022), from high data center electricity consumption to the difficulty of integrating models into mobile (Cai et al., 2022) or embedded devices (Peccia & Bringmann, 2024).

Thus, model optimization has become a critical challenge (Kallimani et al., 2023; Sanh et al., 2019; Kurtic et al., 2022). Reducing computational requirements without sacrificing quality is essential for accessibility, ecological sustainability, and practical deployment (Patterson et al., 2022; Wu et al., 2021; Shoukourian et al., 2017; Osondu, 2025; Vanu et al., 2024; Li et al., 2023). Proposed strategies include quantization (Gholami et al., 2022; Liu et al., 2021; Lin et al., 2021; Xiao et al., 2022), weight factorization (Chin et al., 2020; Sainath et al., 2013; Hu et al., 2021; Hao et al., 2024), low-bitwidth representations (Wang et al., 2022; Simons & Dah-Jye, 2019; Dettmers & Zettlemoyer, 2022), and specialized hardware (Reuther et al., 2021; Burhanuddin, 2023; Tuli & Jha, 2023). However, many approaches face trade-offs in universality, complexity, or accuracy. Among the most promising directions is pruning (Cheng et al., 2024; Sundar & Dwaraknath, 2021; Frantar & Alistarh, 2023; Gao et al., 2022; Li et al., 2016; He et al., 2017; Zafrir et al., 2021), which simplifies networks by removing redundant parameters. Beyond engineering gains, pruning provides insights into network structure and has proven effective across image classification (Bai et al., 2023; Tang et al., 2022; Pan et al., 2022), text processing (Ma et al., 2023; Kurtic et al., 2023; Shim et al., 2021), and generative models (Saxena et al., 2024; Brahim Belhaouari & Kraidia, 2025; Kafle et al., 2025), achieving significant efficiency improvements.

Despite its advantages, pruning still suffers from heuristic reliance, poor scalability, and limited ability to capture information propagation dynamics (Cheng et al., 2024; Sundar & Dwaraknath, 2021; Frantar & Alistarh, 2023; Gao et al., 2022; Li et al., 2016; He et al., 2017; Zafrir et al., 2021; Bai et al., 2023; Tang et al., 2022; Pan et al., 2022; Ma et al., 2023; Kurtic et al., 2023; Shim et al., 2021; Saxena et al., 2024; Brahim Belhaouari & Kraidia, 2025; Kafle et al., 2025). To address this, we propose a two-stage optimization framework based on the concept of information flow divergence, a formal metric quantifying signal evolution through layers.

The first stage targets filter-level optimization: divergence measurements (Dineen, 2014; Tran, 2018; Perrella et al., 2023; Lopes & Ruggiero, 2021; Kim et al., 2013; Machenhauer & Rasmussen, 1972; Rezende & Mohamed, 2016) prune redundant parameters while preserving critical pathways (Shwartz-Ziv, 2022; Saxe et al., 2018; Wu et al., 2022; Munezero et al., 2021; Yu et al., 2025; Greff et al., 2015). The second stage extends to layer-level compression, consolidating blocks based on their contribution to overall information throughput. Unlike traditional methods that focus only on parameter or layer counts, our framework jointly optimizes both while respecting information dynamics.

We provide algorithmic specifications for various layer types and demonstrate that this holistic approach outperforms isolated strategies. Experiments across convolutional and transformer architectures show substantial model size reductions without compromising functionality.

Ultimately, this framework is not only a compression tool but a new perspective on neural network design, where measurable information flow guides architectural decisions, enabling models that are smaller and computationally more efficient.

Thus, the main contributions of our work to neural network compression are as follows:

- **Two-Stage Holistic Compression Framework**. We propose the first pruning methodology that systematically optimizes neural networks along both *width* (filter-level) and *depth* (layer-level) dimensions through a unified flow-divergence criterion. The framework combines:
  - Stage 1: *Divergence-Aware Filter Pruning* (IDAP).
  - Stage 2: *Flow-Guided Layer Truncation*.
- **Theory of Information Flow Divergence**. A mathematically rigorous formulation of neural network dynamics as continuous signal propagation systems, with:
  - Integral-based divergence measures for discrete/continuous layers.
  - Architecture-agnostic flow conservation principles.
- **Computational Machinery**:
  - Efficient algorithms for flow computation in FC/Conv/Attention layers ($O(L)$ complexity).

  - Adaptive thresholding for joint filter-layer optimization.
- **Empirical Validation**:
  - $\sim$75-90% CNN pruning with $<$2% accuracy drop.
  - $>$70% transformers pruning while maintaining $\sim$98%+ baseline accuracy.
  - $>$40% faster inference post-compression.

2 Problem Statement

Modern neural networks are heavily overparameterized, with many operations contributing little to performance and adding unnecessary complexity (Morcos et al., 2018).

The key challenge is to reduce this complexity while preserving accuracy, robustness, generalization, and adaptability across tasks such as classification, text generation, and image synthesis. This is complicated by heterogeneous architectures, intricate internal dynamics, and the limited interpretability of pruning effects. Scaling optimization methods to large models further demands high efficiency.

These factors underscore the need for principled approaches that can reliably detect redundancy and optimize structures while accounting for internal information processes. In this work, we address this problem with a pruning framework grounded in information flow dynamics, which enables the safe removal of non-essential components.

3 Proposed Solution

3.1 Information Flow Dynamics in Deep Neural Networks

We present a comprehensive theoretical framework for analyzing information propagation through deep neural networks by modeling them as dynamical systems that transform input data through successive nonlinear transformations. The key insight is to characterize how information content evolves as it flows through the network's computational path.

3.1.1 Continuous Flow Representation

For a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters θ, we represent its computations as a continuous trajectory:

$$\mathbf{T}(s) = f_\theta(\mathbf{x}, s), \quad s \in [0, 1], \tag{1}$$

where:

- $s = 0$ corresponds to the input layer;
- $s = 1$ corresponds to the output layer;
- intermediate s values represent hidden transformations.

The differential change captures the instantaneous information flow:

$$\phi(s) = \frac{d\mathbf{T}}{ds}(s) = \lim_{\Delta s \to 0} \frac{\mathbf{T}(s + \Delta s) - \mathbf{T}(s)}{\Delta s}. \tag{2}$$

This formulation offers several important advantages. First, it establishes a connection to dynamical systems theory, providing a solid mathematical foundation for analyzing information flow. Second, it enables a unified treatment of both discrete and continuous architectures. Finally, it naturally accommodates residual connections.

3.1.2 Flow Divergence Measure

We define flow divergence to quantify information dissipation/concentration:

$$\mathcal{D}(s) = \frac{d^2\mathbf{T}}{ds^2}(s) \cdot \left(\frac{d\mathbf{T}}{ds}(s) \right)^\top. \tag{3}$$

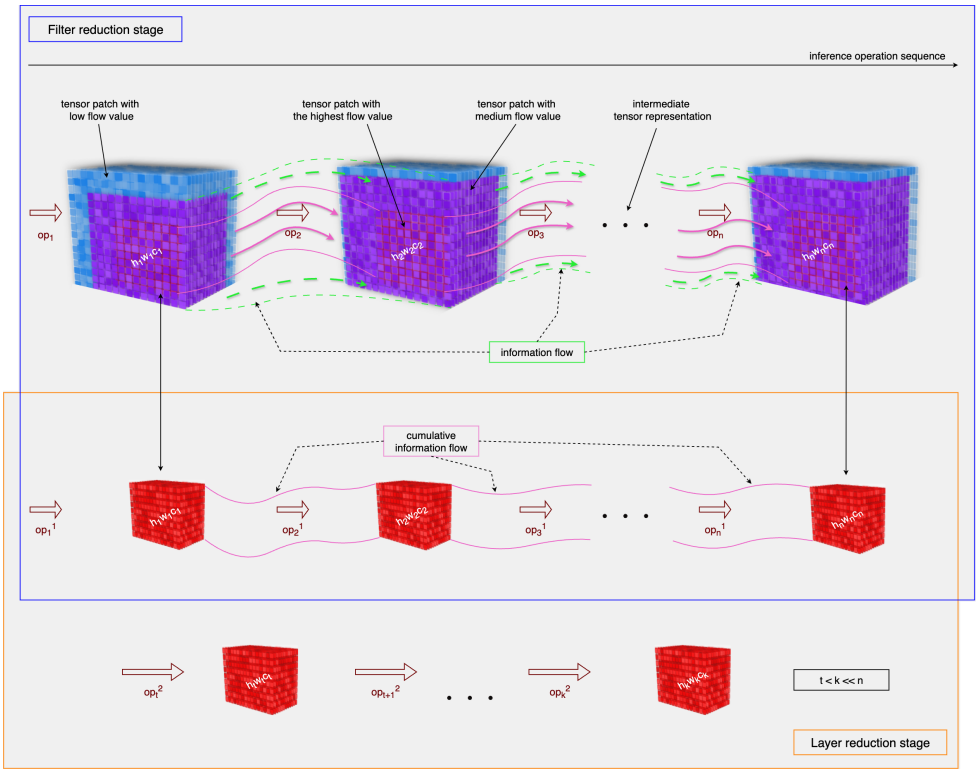

Figure 1: Visualization of information flow through network depth. Arrows represent derivative-based flow measurements at different depth coordinates s.

For practical computation in discrete networks with L layers:

$$\mathcal{D}_l = \underbrace{\frac{\|\mathbf{T}_{l+1} - \mathbf{T}_l\|_2}{\|\mathbf{T}_l\|_2 + \epsilon}}_{\text{Relative change}} \cdot \underbrace{\big|\|\mathbf{W}_{l+1}\mathbf{T}_l\|_2 - \|\mathbf{W}_l\mathbf{T}_{l-1}\|_2\big|}_{\text{Weighted transformation difference}}, \tag{4}$$

where $\epsilon = 10^{-6}$ prevents numerical instability. This approximation preserves derivative-based interpretation and remains computationally tractable. It also captures both magnitude and directional changes.

We also provide an extension of the flow divergence measure through variance-based normalization (see Section A.1), which improves interpretability and robustness compared to exponential normalization. Furthermore, we present a formal treatment of the key mathematical properties of the introduced divergence measure (see Section A.2), including scale invariance and additive composition.

Now we formalize a two-stage (the order and mechanics of the stages are determined empirically according to our experiments) algorithm **IDAP++**. At the first stage, we eliminate insignificant filters, and at the second stage, we remove insignificant layers. In this case, the criteria of significance are determined through the above-introduced concept of divergence of the information flow inside the neural network (Fig. 1).

3.2 Compression Stage 1: Filters Reduction

Building upon the flow divergence framework established in Section 3.1, we now present the first stage of our compression pipeline: structured filter pruning guided by information flow analysis. This stage operates at the granularity of individual filters or attention heads, removing those that contribute minimally to the network's information throughput while preserving critical pathways.

To begin with, we formalize the concept of divergence for the most fundamental types of layers in neural networks (Section B).

For fully connected layers, we define divergence in terms of the Jacobian sensitivity, activation norm, and weight norm, showing how their interaction reflects both the responsiveness and structural importance of the layer (Section B.1). For convolutional layers, we extend the formulation to activation tensors and convolutional kernels, incorporating normalization by activation volume and demonstrating adaptability to architectural variations (Section B.2). For self-attention layers, we derive both single-head and multi-head divergence measures, decomposing the role of query/key/value projections and attention patterns, and proving additive composition across heads (Section B.3).

Within the scope of this study, we formulate the principles of divergence computation for different neural network architectures comprising various types of layers. All related materials are presented in a dedicated section C, which includes step-by-step algorithms for divergence computation, accompanied by an analysis of their algorithmic complexity and an assessment of computational overhead. In particular, separate subsections address fully connected architectures (see C.1), convolutional architectures (see C.2), and attention-based architectures (see C.3).

Now, let us introduce a generalized pruning methodology that systematically removes network parameters while preserving information flow characteristics in the **Iterative Divergence-Aware Pruning (IDAP)** technique. A step-by-step detailed procedure is presented in Section D (Algorithm 5).

The method exhibits several key features. First, it employs progressive sparsification, where the pruning ratio ρ_k increases non-linearly with iteration k, controlled by a scaling parameter α. Second, the pruning process is guided by divergence, removing weights with the highest flow divergence scores $\mathcal{D}$. Additionally, the procedure incorporates a performance-aware termination criterion, ceasing further pruning when the drop in validation accuracy exceeds a predefined threshold τ. Finally, the algorithm is capable of automatically selecting the optimal pruning ratio ρ^* from among the tested configurations.

The implementation relies on layer-specific divergence computations as described in Sections C.1–C.3. Fine-tuning is performed using the original training schedule but with a reduced learning rate to stabilize the pruned model. The pruning aggressiveness is governed by the parameter α, which is typically selected from the range 0.5 to 2.0.

3.3 Stage 2: Flow-Guided Layer Truncation

After filter pruning, our method eliminates layers strategically via information flow analysis, removing those with minimal contribution to information propagation while maximizing error reduction. The step-by-step procedure is outlined in the corresponding Section E (Algorithm 6).

The proposed method relies on two core components: information flow scoring and an adaptive replacement strategy.

Information Flow Scoring quantifies the relative contribution of each layer l by computing its normalized flow divergence across the validation set:

$$\mathcal{D}_l = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{val}}} \frac{\|\mathbf{T}_{l+1}(\mathbf{x}) - \mathbf{T}_l(\mathbf{x})\|_2}{\|\mathbf{T}_l(\mathbf{x})\|_2 + \epsilon},$$

where $\mathbf{T}_l(\mathbf{x})$ denotes the output of layer l for input $\mathbf{x}$.

Adaptive Replacement Strategy ensures that structurally important components are preserved while enabling architectural simplification. It combines identity and projection mappings to maintain dimensional compatibility (denoted as Identity* Mapping), applies local fine-tuning to adjacent layers for stability, and uses error-driven selection to prioritize replacements that yield the greatest reduction in validation loss, denoted δE.

3.4 IDAP++: Unified Two-Stage Compression Framework

IDAP++ Algorithm 1 implements a two-stage compression methodology that progressively removes redundant components while preserving information flow.

The proposed framework exhibits several key features. It ensures a *seamless transition* from filter pruning to layer removal by incorporating intermediate recomputation of information flow. Both stages rely on a *unified flow metric*, using a consistent divergence measure:

$$\mathcal{D}_l = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\text{val}}}\left[\frac{\|\mathbf{T}_{l+1}(\mathbf{x}) - \mathbf{T}_l(\mathbf{x})\|_2}{\|\mathbf{T}_l(\mathbf{x})\|_2 + \epsilon}\right].$$

The method also introduces *adaptive budget allocation*, automatically distributing the total accuracy degradation budget Δ_{max} equally between the two pruning phases, with dynamic adjustment based on actual performance outcomes. Finally, the framework employs *compression-aware fine-tuning*, which includes local tuning of candidate layers during removal, intermediate rebalancing following filter pruning, and global fine-tuning at the final stage to restore performance.

The theoretical validity of this method is supported by the theorem presented below.

Theorem 1. *For any network $\mathcal{N}_0$ compressed with IDAP++, the compressed network $\mathcal{N}^*$ satisfies:*

$$\frac{\|\mathcal{N}_0(\mathbf{x}) - \mathcal{N}^*(\mathbf{x})\|_2}{\|\mathcal{N}_0(\mathbf{x})\|_2} \leq \Delta_{max} \quad \forall \mathbf{x} \in \mathcal{D}_{val}, \tag{5}$$

while achieving maximal sparsity under the given constraints.

Algorithm 1 Integrated IDAP++ Compression Pipeline

Require:
- Initial network $\mathcal{N}_0$ with parameters Θ
- Validation dataset $\mathcal{D}_{\text{val}}$
- Target accuracy drop Δ_{max}
- Pruning hyperparameters α, β

Ensure: Compressed network $\mathcal{N}^*$

Initialize compression tracker: $\mathcal{C} \leftarrow \{\}$
Compute initial flow: $\mathcal{D} \leftarrow \text{ComputeFlowDivergence}(\mathcal{N}_0, \mathcal{D}_{\text{val}})$
Phase 1: Adaptive Filter Pruning
for iteration $t \leftarrow 1$ T_{filter} **do**
  Determine pruning threshold: $\tau_t \leftarrow \text{Percentile}(\mathcal{D}, p_0(1 + t/T_{\text{filter}})^\alpha)$
  Generate pruning mask: $\mathbf{M}_t \leftarrow \mathbb{I}[\mathcal{D} > \tau_t]$
  Evaluate compressed network: $\mathcal{N}_t \leftarrow \mathcal{N}_{t-1} \odot \mathbf{M}_t$ $\text{Acc}_t \leftarrow \text{Validate}(\mathcal{N}_t, \mathcal{D}_{\text{val}})$
  if $\text{Acc}_0 - \text{Acc}_t > \Delta_{\text{max}}/2$ **then**
    Revert to $\mathcal{N}_{t-1}$
    break
  end if
  Update compression tracker: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(t, \|\mathbf{M}_t\|_0)\}$
end for
Phase Transition: Flow Rebalancing
$\mathcal{N}_{\text{inter}} \leftarrow \text{IntermediateFineTune}(\mathcal{N}_t)$
Recompute flow: $\mathcal{D}' \leftarrow \text{RecomputeFlowDivergence}(\mathcal{N}_{\text{inter}}, \mathcal{D}_{\text{val}})$
Phase 2: Strategic Layer Removal
for layer l in $\text{SortLayersByFlow}(\mathcal{D}')$ **do**
  Create candidate network: $\mathcal{N}_{\text{cand}} \leftarrow \text{ReplaceLayer}(\mathcal{N}_{\text{inter}}, l, \text{Identity})$
  Local fine-tuning: $\mathcal{N}_{\text{cand}} \leftarrow \text{AdaptiveFineTune}(\mathcal{N}_{\text{cand}}, \text{Neighborhood}(l))$
  if $\text{Acc}_0 - \text{Validate}(\mathcal{N}_{\text{cand}}, \mathcal{D}_{\text{val}}) < \Delta_{\text{max}}$ **then**
    Accept removal: $\mathcal{N}_{\text{inter}} \leftarrow \mathcal{N}_{\text{cand}}$
    Update tracker: $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{Removed } l\}$
  end if
  if $\text{Acc}_0 - \text{Validate}(\mathcal{N}_{\text{inter}}, \mathcal{D}_{\text{val}}) > \Delta_{\text{max}}$ **then**
    break
  end if
end for
return $\mathcal{N}^* \leftarrow$ **GlobalFineTune**$(\mathcal{N}_{\text{inter}}, \mathcal{D}_{\text{val}}), \mathcal{C}$

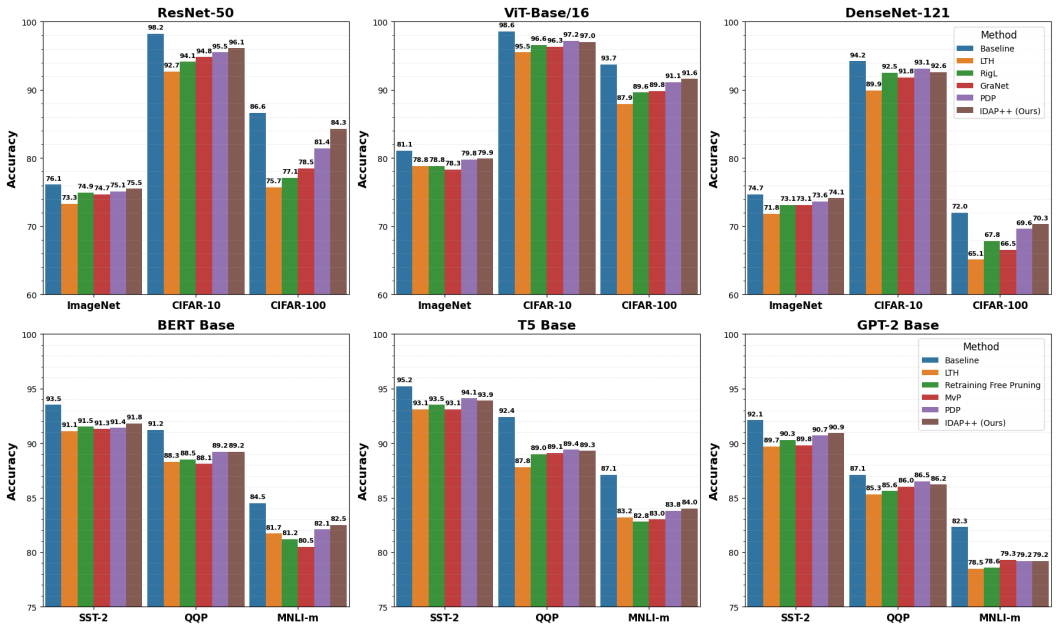

Figure 2: Comparison of pruning methods under 50-80% sparsity.

4 Experimental Setup and Results

As part of this study, we developed a comprehensive experimental platform designed for an evaluation of the proposed iterative pruning method for neural networks, incorporating information flow characteristics into the optimization process. The platform provides a unified environment for objective comparison of optimization results across diverse model architectures and datasets and for assessing the impact of progressive parameter removal on key performance indicators.

The experimental infrastructure consists of three interconnected functional components. The first component is a module for calculating flow characteristics, which automatically determines the contribution of each network layer to information transmission and transformation. This serves as the basis for informed structural simplification decisions. The second component is an intelligent optimization mechanism that implements stepwise reduction of neural network parameters with dynamic control over the impact of pruning on model accuracy. The third component is a standardized testing module that ensures reproducibility and comparability of experiments across various neural architectures, including classical convolutional networks and modern transformer-based models.

To comprehensively evaluate the proposed approach, we selected a range of widely used neural network architectures from computer vision. Our experiments included classification models such as ResNet-50 (He et al., 2015), EfficientNet-B4 (Tan & Le, 2019b), ViT-Base/16 (Dosovitskiy et al., 2021), MobileNetV3-Large (Howard et al., 2019), DenseNet-121 (Huang et al., 2017), ConvNeXt-Small (Liu et al., 2022), VGG19-BN (Simonyan & Zisserman, 2014), and ShuffleNet V2 x2.0 (Ma et al., 2018). We also used object detection and image segmentation models, including Faster R-CNN (Ren et al., 2015), YOLOv4 (Bochkovskiy et al., 2020), DETR (Carion et al., 2020), FCN (Long et al., 2015), U-Net (Ronneberger et al., 2015), and SegFormer (Xie et al., 2021). Furthermore, we tested generative architectures such as DCGAN (Radford et al., 2015), VQGAN (Esser et al., 2021), and Stable Diffusion v1.5 (Rombach et al., 2022).

To validate the generality of our pruning method, we extended the evaluation to other modalities, specifically natural language processing (NLP), using BERT Base (Devlin et al., 2019), GPT-2 Base (Radford et al., 2019), and T5 Base (Raffel et al., 2020).

Testing was performed on various benchmark datasets representing a diverse range of computer vision and NLP tasks: ImageNet (Deng et al., 2009), CIFAR-10 (Krizhevsky et al., 2009), CIFAR-

100 (Krizhevsky et al., 2009), Stanford Cars (Krause et al., 2013), Flowers-102 (Nilsback & Zisserman, 2008), iNaturalist (Van Horn et al., 2018), Food101 (Bossard et al., 2014), Oxford-IIIT Pet (Parkhi et al., 2012), Fashion MNIST (Xiao et al., 2017), FER2013 (Carrier & Courville, 2013), Pascal VOC (Everingham et al., 2010), COCO 2017 (Lin et al., 2014), COCO-Stuff (Caesar et al., 2018), MNLI-m (Wang et al., 2018), SQuAD 1.1 (Rajpurkar et al., 2016) and other datasets.

For each combination of architecture and dataset, the system automatically calculates layer-specific flow metrics, followed by iterative pruning with a nonlinear increase in pruning intensity. This approach allows precise control over the trade-off between model simplification and preservation of functional performance. At each pruning step, the compressed model's performance is validated, and the process continues until the predefined threshold for acceptable accuracy degradation is reached.

Throughout the experiments, four key metrics are recorded: the percentage of removed weights relative to the original parameter count, the remaining model accuracy on the test set, the absolute accuracy drop compared to the baseline model, and the estimated reduction in computational complexity, expressed through floating-point operation counts (FLOPs).

A detailed comparison of pruning results across different architectures and datasets is provided in Table 1 and Fig. 2. The results demonstrate that IDAP++ achieves significant computational reductions, with FLOPs typically decreasing by 57–75% and model parameters by 67-69% for language models. While accuracy drops were generally moderate for vision models (mostly within 1–4%), generative models and language models exhibited more pronounced sensitivity, with FID scores increasing by 7–9% and accuracy dropping by 4–5%. For example, on image classification tasks, ViT-Base/16 on CIFAR-10 retained 97.0% accuracy with a 75% FLOPs reduction. In contrast, architectures like ShuffleNetV2 and language models like BERT and GPT-2 showed greater sensitivity to pruning.

Additionally, Fig. 2 provides a comparative analysis of the proposed pruning method against state-of-the-art alternatives on different tasks and benchmarks. IDAP++ consistently outperformed the most common state-of-the-art architectures, including LTH (Frankle & Carbin, 2019), RigL (Evci et al., 2020), GraNet (Wang et al., 2023), PDP (Cho et al., 2023), Retraining Free Pruning (Kwon et al., 2022), and MvP (Sanh et al., 2020) under 50-80% sparsity.

Table 1: Pruning results for different architectures using IDAP++

Architecture	**Dataset**	**Metric**				**Model Size**			
		Name	Base	Pruned	Δ%	Name	Base	Pruned	Δ%
ResNet-50	ImageNet	Acc@1	76.1	74.6	-2.0	GFlops	4.1	1.5	-63
EfficientNet-B4	CIFAR-100	Acc@1	90.1	88.1	-2.3	GFlops	4.2	1.5	-65
ViT-Base/16	CIFAR-10	Acc@1	98.6	97.0	-1.6	GFlops	17.5	4.3	-75
Faster R-CNN (ResNet-50)	Pascal VOC	mAP	78.4	76.7	-4.1	GFlops	150	62	-59
YOLOv4 (ShuffleNetV2)	Pascal VOC	mAP	77.5	75.8	-4.1	GFlops	52	22	-58
DETR (ViT-Base/16)	COCO 2017	mAP	42.0	40.5	-3.6	GFlops	87	36	-57
FCN (VGG19-BN)	Cityscapes	mIoU	70.2	68.9	-1.9	GFlops	213	83	-61
U-Net (ResNet-50)	Pascal VOC	mIoU	75.8	74.2	-2.1	GFlops	170	62	-64
SegFormer (ViT-Base/16)	COCO 2017	mIoU	47.0	45.1	-4.0	GFlops	163	63	-61
DCGAN	CIFAR-10	FID	24.1	25.9	+6.9	GFlops	12.2	4.8	-61
VQGAN	COCO-Stuff	FID	18.5	20.1	+8.0	GFlops	18.3	7.5	-59
Stable Diffusion v1.5	MS-COCO	FID	12.3	13.5	+8.9	GFlops	86	34	-60
BERT Base	MNLI-m	Acc	84.5	82.5	-5.4	Params (M)	110	37	-67
GPT-2 Base	SQuAD 1.1	F1	86.3	82.6	-4.3	Params (M)	117	36	-69
T5 Base	MNLI-m	Acc	87.1	83.7	-3.9	Params (M)	220	71	-68

We have also included some complementary experimental results in Section F. Table 3 demonstrates the dynamics of model compression applied to ResNet-50 over 35 pruning iterations on CIFAR-10. The gradual pruning reduced GFLOPs from 4.09 to 1.14 (a nearly 72% decrease), while Top-1 accuracy decreased from 98.20% to 95.98%. The table highlights that accuracy remained above 97% for more than 25 pruning steps, with sharper drops only in the final layer truncation stages. This highlights the robustness of IDAP++ in maintaining high performance under aggressive compression.

A separate comparison of inference time for the aforementioned architectures was conducted, with the results presented in Table 4. Pruning achieved notable acceleration across all models, with speedups ranging from 1.50× (GPT-2 Base) to 2.16× (MobileNetV3-L). Lightweight architectures such as ShuffleNetV2 and MobileNetV3 benefited the most, while heavier models like ViT and ConvNeXt showed more modest gains.

It should also be noted that repeated application of the algorithm did not preserve acceptable accuracy while significantly reducing the number of model parameters.

We have made our implementation publicly available on GitHub (Author, 2025) to ensure reproducibility and facilitate further research. More detailed and comprehensive results of pruning various architectures across different modalities and benchmarks using IDAP++ are also available in the GitHub repository (Author, 2025).

5 Discussions and Conclusion

The growing need to reduce parameters and simplify neural network topologies while preserving information semantics has intensified interest in compression techniques. This work introduces a theoretically grounded, two-stage framework addressing redundancy at both filter and architecture levels. Central to our approach is the formalization of information flow dynamics, a metric quantifying signal evolution across layers and bridging information theory with practical compression.

Our method builds on the tensor flow divergence concept, adapted from continuum mechanics for discrete graphs. Experiments across a wide spectrum of models (from convolutional networks and vision transformers to generative models and natural language processing architectures like BERT and GPT-2) confirm that many parameters are redundant and can be safely removed. Notably, filter pruning (Stage 1) and layer truncation (Stage 2) act complementarily: width reduction simplifies depth optimization by eliminating noisy pathways. The flow divergence metric also shows task-robust consistency across different modalities.

Beyond performance, the framework provides theoretical insight. The derivative-based flow formulation (dT/ds) suggests neural networks behave as learnable PDEs, where transformation smoothness outweighs parameter count. This explains why our method preserves information coherence where conventional approaches degrade. Still, challenges remain: irregular topologies (e.g., neural ODEs, hypernetworks) and dynamic inputs may require adaptive divergence measures and thresholds. Designing architectures with built-in compressibility emerges as a promising direction.

Practically, our method enables major efficiency gains. On CIFAR-10, ResNet-50 achieves $\sim$80% FLOPs reduction with only $\sim$2% accuracy drop, reclaiming 70–85% of computational budgets typical for large models. For language models, the method achieved a parameter reduction of 67–69%, demonstrating its significant potential for deploying large-scale NLP applications in resource-constrained environments. Such results highlight that efficiency stems not from parameter volume but from the organization of information pathways.

Looking ahead, two research paths are most promising: (i) integration of flow-aware pruning with quantization, and (ii) hardware-sensitive divergence metrics for co-design.

In conclusion, reframing neural networks as information flow systems reveals their essential computational skeletons. The successful application of our method to both vision and language tasks underscores the broad applicability of the information flow principle. This work contributes not just a compression tool but a conceptual framework where efficiency follows from fundamental laws of signal propagation, paving the way for accessible, sustainable, and high-performance AI.

REFERENCES

Unknown Author. Idap++: Advancing divergence-aware pruning with joint filter and layer optimization. https://github.com/user534440/idap_plus_plus, 2025.

Shipeng Bai, Jun Chen, Xintian Shen, Yixuan Qian, and Yong Liu. Unified data-free compression: Pruning and quantization without fine-tuning, 2023. URL https://arxiv.org/abs/2308.07209.

Luciano Baresi and Giovanni Quattrocchi. *Training and Serving Machine Learning Models at Scale*, pp. 669–683. 11 2022. ISBN 978-3-031-20983-3. doi: 10.1007/978-3-031-20984-0_48.

Liane Bernstein, Alexander Sludds, Ryan Hamerly, Vivienne Sze, Joel Emer, and Dirk Englund. Freely scalable and reconfigurable optical hardware for deep learning. *Scientific Reports*, 11, 02 2021. doi: 10.1038/s41598-021-82543-3.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pp. 446–461. Springer, 2014.

Samir Brahim Belhaouari and Insaf Kraidia. Efficient self-attention with smart pruning for sustainable large language models. *Scientific Reports*, 15, 03 2025. doi: 10.1038/s41598-025-92586-5.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and Dario Amodei. Language models are few-shot learners, 05 2020.

M.A. Burhanuddin. Efficient hardware acceleration techniques for deep learning on edge devices: A comprehensive performance analysis. *KHWARIZMIA*, 2023:1–10, 08 2023. doi: 10.70470/KHWARIZMIA/2023/010.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218, 2018.

Han Cai, Ji Lin, Yujun Lin, Zhijian Liu, Haotian Tang, Hanrui Wang, Ligeng Zhu, and Song Han. Enable deep learning on mobile devices: Methods, systems, and applications. *ACM Transactions on Design Automation of Electronic Systems*, 27(3):1–50, March 2022. ISSN 1557-7309. doi: 10.1145/3486618. URL http://dx.doi.org/10.1145/3486618.

James Cameron, Alexandra Sala, Georgios Antoniou, Paul Brennan, Holly Butler, Justin Conn, Siobhan Connal, Tom Curran, Mark Hegarty, Rose McHardy, Daniel Orringer, David Palmer, Benjamin Smith, and Matthew Baker. Multi-cancer early detection with a spectroscopic liquid biopsy platform, 05 2022.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

Pierre-Luc Carrier and Aaron Courville. Fer-2013 dataset. https://www.kaggle.com/datasets/msambare/fer2013, 2013.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):10164–10183, December 2024. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3435937. URL https://doi.org/10.1109/TPAMI.2024.3435937.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):10558–10578, December 2024. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3447085. URL https://doi.org/10.1109/TPAMI.2024.3447085.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. pp. 1515–1525, 06 2020. doi: 10.1109/CVPR42600.2020.00159.

Minsik Cho, Saurabh Adya, and Devang Naik. Pdp: Parameter-free differentiable pruning is all you need. *Advances in Neural Information Processing Systems*, 36:45833–45855, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yogita Desai. Diagnosis of medical images using convolutional neural networks. *Journal of Electrical Systems*, 20:2371–2376, 05 2024. doi: 10.52783/jes.3220.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws, 12 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Seán Dineen. *The Divergence Theorem*, pp. 179–191. Springer London, London, 2014. ISBN 978-1-4471-6419-7. doi: 10.1007/978-1-4471-6419-7_15.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, pp. 2943–2952. PMLR, 2020.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

Shangqian Gao, Feihu Huang, Yanfu Zhang, and Heng Huang. *Disentangled Differentiable Network Pruning*, pp. 328–345. 11 2022. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_20.

Amir Gholami, Sehoon Kim, Dong Zhen, Zhewei Yao, Michael Mahoney, and Kurt Keutzer. *A Survey of Quantization Methods for Efficient Neural Network Inference*, pp. 291–326. 01 2022. ISBN 9781003162810. doi: 10.1201/9781003162810-13.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Klaus Greff, Rupesh Srivastava, Jan Koutník, Bas Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28, 03 2015. doi: 10.1109/TNNLS.2016.2582924.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: low-rank adapters are secretly gradient compressors. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1398–1406, 2017. doi: 10.1109/ICCV.2017.155.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Giorgos Iacovides, Thanos Konstantinidis, Mingxue Xu, and Danilo Mandic. Finllama: Llm-based financial sentiment analysis for algorithmic trading. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, pp. 134–141, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710810. doi: 10.1145/3677052.3698696. URL https://doi.org/10.1145/3677052.3698696.

Andrey Ignatov, Radu Timofte, Shuai Liu, Chaoyu Feng, Furui Bai, Xiaotao Wang, Lei Lei, Ziyao Yi, Yan Xiang, Zibin Liu, Shaoqing Li, Keming Shi, Dehui Kong, Ke Xu, Minsu Kwon, Yaqi Wu, Jiesi Zheng, Zhihao Fan, Xun Wu, and Mingxuan Cai. *Learned Smartphone ISP on Mobile GPUs with Deep Learning, Mobile AI AIM 2022 Challenge: Report*, pp. 44–70. 02 2023. ISBN 978-3-031-25065-1. doi: 10.1007/978-3-031-25066-8_3.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Yu Jiang, Wei Wang, and Chunhui Zhao. A machine vision-based realtime anomaly detection method for industrial products using deep learning. In *2019 Chinese Automation Congress (CAC)*, pp. 4842–4847, 2019. doi: 10.1109/CAC48633.2019.8997079.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon Kohl, Andrew Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 07 2021. doi: 10.1038/s41586-021-03819-2.

Swatantra Kafle, Geethu Joseph, and Pramod K. Varshney. One-bit compressed sensing using generative models, 2025. URL https://arxiv.org/abs/2502.12762.

Rakhee Kallimani, Krishna Pai, Prasoon Raghuwanshi, Sridhar Iyer, and Onel Alcaraz López. Tinyml: Tools, applications, challenges, and future research directions. *Multimedia Tools and Applications*, 09 2023. doi: 10.1007/s11042-023-16740-9.

Kitae Kim, Soohyun Cho, and Woojin Chung. Hd map update for autonomous driving with crowdsourced data. *IEEE Robotics and Automation Letters*, PP:1–1, 02 2021. doi: 10.1109/LRA.2021.3060406.

Yusik Kim, Ian Castro, and Zheng-Tong Xie. Divergence-free turbulence inflow conditions for large-eddy simulations with incompressible flow solvers. *Computers and Fluids*, 84, 09 2013. doi: 10.1016/j.compfluid.2013.06.001.

Volodymyr Kindratenko, Robert Wilhelmson, Robert Brunner, Todd Martinez, and Wen-mei Hwu. High-performance computing with accelerators. *Computing in Science Engineering*, 12:12 – 16, 09 2010. doi: 10.1109/MCSE.2010.88.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. pp. 4163–4181, 01 2022. doi: 10.18653/v1/2022.emnlp-main.279.

Eldar Kurtic, Elias Frantar, and Dan Alistarh. Ziplm: inference-aware structured pruning of language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116, 2022.

JunKyu Lee, Lev Mukhanov, Amir Sabbagh Molahosseini, Umar Minhas, Yang Hua, Jesus Martinez del Rincon, Kiril Dichev, Cheol-Ho Hong, and Hans Vandierendonck. Resource-efficient convolutional networks: A survey on model-, arithmetic-, and implementation-level techniques. *ACM Comput. Surv.*, 55(13s), July 2023. ISSN 0360-0300. doi: 10.1145/3587095. URL https://doi.org/10.1145/3587095.

Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Clover: Toward sustainable ai with carbon-aware machine learning inference service. pp. 1–15, 11 2023. doi: 10.1145/3581784.3607034.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and H.P. Graf. Pruning filters for efficient convnets. 08 2016. doi: 10.48550/arXiv.1608.08710.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Fully quantized vision transformer without retraining, 11 2021.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A review on background, technology, limitations, and opportunities of large vision models, 2024a. URL https://arxiv.org/abs/2402.17177.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases, 2024b. URL https://arxiv.org/abs/2402.14905.

Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer, 2021. URL https://arxiv.org/abs/2106.14156.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Artur Lopes and Rafael Ruggiero. Nonequilibrium in thermodynamic formalism: the second law, gases and information geometry, 03 2021.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023. URL https://arxiv.org/abs/2305.11627.

Bennert Machenhauer and E. Rasmussen. On the integration of the spectral hydrodynamical equations by a transform method. 01 1972.

Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of dall-e 2, 2022. URL https://arxiv.org/abs/2204.13807.

Scott McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona Gilbert, Mark Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher Kelly, Dominic King, and Shravya Shetty. Addendum: International evaluation of an ai system for breast cancer screening. *Nature*, 586:E19–E19, 10 2020. doi: 10.1038/s41586-020-2679-9.

Amil Merchant, Simon Batzner, Samuel Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Cubuk. Scaling deep learning for materials discovery. *Nature*, 624:1–6, 11 2023. doi: 10.1038/s41586-023-06735-9.

Santiago Miret, N M Anoop Krishnan, Benjamin Sanchez, Marta Skreta, Vineeth Venugopal, and Jennifer Wei. Perspective on ai for accelerated materials design at the ai4mat-2023 workshop at neurips 2023. *Digital Discovery*, 3, 05 2024. doi: 10.1039/d4dd90010c.

Ari S. Morcos, David G. T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization, 2018. URL https://arxiv.org/abs/1803.06959.

Yao Mu, Shoufa Chen, Mingyu Ding, Jianyu Chen, Runjian Chen, and Ping Luo. Ctrlformer: Learning transferable state representation for visual control via transformer, 2022. URL https://arxiv.org/abs/2206.08883.

Parfait Munezero, Mattias Villani, and Robert Kohn. Dynamic mixture of experts models for online prediction, 09 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and Barret Zoph. Gpt-4 technical report, 03 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Joshua Osondu. Red ai vs. green ai in education: How educational institutions and students can lead environmentally sustainable artificial intelligence practices, 01 2025.

Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. *EdgeViTs: Competing Light-Weight CNNs on Mobile Devices with Vision Transformers*, pp. 294–311. 11 2022. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_18.

Omkar M Parkhi, Andrea Vedaldi, et al. Cats and dogs. *CVPR*, 2012. URL https://www.robots.ox.ac.uk/~vgg/data/pets/.

David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluís-Miquel Munguía, Daniel Rothchild, David So, Maud Texier, and Jeffrey Dean. The carbon footprint of machine learning training will plateau, then shrink, 02 2022.

Federico Nicolás Peccia and Oliver Bringmann. Embedded distributed inference of deep neural networks: A systematic review, 2024. URL https://arxiv.org/abs/2405.03360.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4, 2023. URL https://arxiv.org/abs/2304.03277.

David Perrella, Nathan Duignan, and David Pfefferlé. Existence of global symmetries of divergence-free fields with first integrals. *Journal of Mathematical Physics*, 64, 05 2023. doi: 10.1063/5.0152213.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Ai accelerator survey and trends. pp. 1–9, 09 2021. doi: 10.1109/HPEC49654.2021.9622867.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

Vladimir Rodriguez-Caballero and Mauricio Villanueva-Domínguez. Predicting cryptocurrency crash dates. *Empirical Economics*, 63:1–19, 03 2022. doi: 10.1007/s00181-022-02229-1.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2022. doi: 10.1109/CVPR52688.2022.01042.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Tara Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. pp. 6655–6659, 10 2013. doi: 10.1109/ICASSP.2013.6638949.

Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, April 2023. ISSN 2377-3774. doi: 10.1109/lra.2023.3246839. URL http://dx.doi.org/10.1109/LRA.2023.3246839.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 10 2019.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20378–20389, 2020. URL https://arxiv.org/abs/2005.07683. arXiv:2005.07683.

Andrew Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Tracey, and David Cox. On the information bottleneck theory of deep learning. 02 2018.

Divya Saxena, Jiannong Cao, Jiahao Xu, and Tarun Kulshrestha. Rg-gan: dynamic regenerative pruning for data-efficient generative adversarial networks. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i5.28271. URL https://doi.org/10.1609/aaai.v38i5.28271.

Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model, 2023. URL https://arxiv.org/abs/2310.15110.

Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. Layer-wise pruning of transformer attention heads for efficient language modeling, 2021. URL https://arxiv.org/abs/2110.03252.

Yeou-Ren Shiue, Ken-Chun Lee, and Chao-Ton Su. Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers Industrial Engineering*, 125, 03 2018. doi: 10.1016/j.cie.2018.03.039.

Hayk Shoukourian, Torsten Wilde, Detlef Labrenz, and Arndt Bode. Using machine learning for data center cooling infrastructure efficiency prediction. pp. 954–963, 05 2017. doi: 10.1109/IPDPSW.2017.25.

Ravid Shwartz-Ziv. Information flow in deep neural networks, 2022. URL https://arxiv.org/abs/2202.06749.

Taylor Simons and Lee Dah-Jye. A review of binarized neural networks. *Electronics*, 8:661, 06 2019. doi: 10.3390/electronics8060661.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Varun Sundar and Rajat Dwaraknath. [reproducibility report] rigging the lottery: Making all tickets winners, 03 2021.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 05 2019a.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019b. URL https://api.semanticscholar.org/CorpusID:167217261.

Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Chao Xu, and Yunhe Wang. Ghostnetv2: enhance cheap operation with long-range attention. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Gemini Team, Google, and Oana David. Gemini: A family of highly capable multimodal models. 12 2023. doi: 10.48550/arXiv.2312.11805.

Gemini Team et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Gemini Team et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Max Tran. Evidence for maxwell's equations, fields, force laws and alternative theories of classical electrodynamics. *European Journal of Physics*, 39, 09 2018. doi: 10.1088/1361-6404/aadf9b.

Charles Edison Tripp, Jordan Perr-Sauer, Jamil Gafur, Amabarish Nag, Avi Purkayastha, Sagi Zisman, and Erik A. Bensen. Measuring the energy consumption and efficiency of deep neural networks: An empirical analysis and design recommendations, 2024. URL https://arxiv.org/abs/2403.08151.

Shikhar Tuli and N.K. Jha. Acceltran: A sparsity-aware accelerator for dynamic inference with transformers, 02 2023.

Kalim Ullah, Hisham Alghamdi, Ghulam Hafeez, Imran Khan, Safeer Ullah, and Sadia Murawwat. A swarm intelligence-based approach for multi-objective optimization considering renewable energy in smart grid. pp. 1–7, 07 2024. doi: 10.1109/ICECET61485.2024.10698431.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Nur Vanu, Salma Akter, and Md Faruque. Legal and ethical frameworks for regulating artificial intelligence in business. *Journal of Business Venturing, AI and Data Analytics*, pp. 1, 08 2024. doi: 10.63471/jbvada24001.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Haowen Wang, Wanhao Niu, and Chungang Zhuang. Granet: A multi-level graph network for 6-dof grasp pose generation in cluttered scenes. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 937–943. IEEE, 2023.

Juliana Wang. Training a convolutional neural network for exoplanet classification with transit photometry data. *Scientific Reports*, 15, 05 2025. doi: 10.1038/s41598-025-98935-8.

Pengyu Wang, Yufan Cheng, Qihang Peng, Binhong Dong, and Shaoqian Li. Low-bitwidth convolutional neural networks for wireless interference identification. *IEEE Transactions on Cognitive Communications and Networking*, 8:557–569, 06 2022. doi: 10.1109/TCCN.2022.3149092.

Felix Wong, Erica Zheng, Jacqueline Valeri, Nina Donghia, Melis Anahtar, Satotaka Omori, Alicia Li, Andres Cubillos-Ruiz, Aarti Krishnan, Wengong Jin, Abigail Manson, Jens Friedrichs, Ralf Helbig, Behnoush Hajian, Dawid Fiejtek, Florence Wagner, Holly Soutter, Ashlee Earl, Jonathan Stokes, and James Collins. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 626:177–185, 12 2023. doi: 10.1038/s41586-023-06887-8.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, and Kim Hazelwood. Sustainable ai: Environmental implications, challenges and opportunities, 10 2021.

Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows, 02 2022.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 11 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions, 2023. URL https://arxiv.org/abs/2306.02224.

Fanghua Yu, Jinjin Gu, Jinfan Hu, Zheyuan Li, and Chao Dong. Unicon: Unidirectional information flow for effective control of large-scale diffusion models, 2025. URL https://arxiv.org/abs/2503.17221.

Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. Prune once for all: Sparse pre-trained language models, 11 2021.

Anita Zarghami. Role of artificial intelligence in surgical decision-making: A comprehensive review: Role of ai in sdm. *Galen Medical Journal*, 13:e3332, 03 2024. doi: 10.31661/gmj.v13i.3332.

Jinjin Zhang, Qiuyu Huang, Junjie Liu, Xiefan Guo, and di Huang. Diffusion-4k: Ultra-high-resolution image synthesis with latent diffusion models, 03 2025.

A Flow Divergence Measure Extensions

A.1 Normalization via Sample Variance

We compute flow statistics using a validation set $\mathcal{D}_{\text{val}} = \{\mathbf{x}_i\}_{i=1}^{N}$ with variance-based normalization:

$$\hat{\mathcal{D}}_l = \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}_l(\mathbf{x}_i) \cdot \left(1 + \frac{\text{Var}(\mathbf{T}_l)}{\sigma^2_{\text{max}}}\right)^{-1}. \tag{6}$$

where:

- $\text{Var}(\mathbf{T}_l)$ is the activation variance across samples;
- σ^2_{max} is the maximum observed variance (for scaling).

This approach offers three benefits over exponential normalization: it provides more interpretable variance scaling, is robust to outlier activations, and preserves layer-wise sensitivity.

A.2 Key Properties of the Introduced Divergence Measure

The divergence measure satisfies two fundamental properties, which are formulated as corresponding lemmas.

Lemma 2 (Scale Invariance)**.** *For any $\alpha > 0$:*

$$\mathcal{D}_l(\alpha \mathbf{T}_l, \alpha \mathbf{T}_{l+1}) = \mathcal{D}_l(\mathbf{T}_l, \mathbf{T}_{l+1}). \tag{7}$$

Lemma 3 (Additive Composition)**.** *For sequential transformations:*

$$\mathcal{D}_{l \to l+2} = \mathcal{D}_l \cdot \mathcal{D}_{l+1} + \mathcal{O}(\|\Delta \mathbf{T}\|^3). \tag{8}$$

B Detailed Divergence Formulation for Different Layer Types

B.1 Divergence Explicit Representation for Fully Connected Layers

Let us first consider the mathematical formulation. For a fully connected layer l with weight matrix $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and activation vector $\mathbf{h}_l \in \mathbb{R}^{n_l}$, the layer-wise divergence $\mathcal{D}_{\text{FC}}^{(l)}$ is computed as:

$$\mathcal{D}_{\text{FC}}^{(l)}(\mathbf{x}) = \underbrace{\|\mathbf{J}(\mathbf{h}_l)\|_F}_{\text{Activation sensitivity}} \cdot \underbrace{\|\mathbf{h}_l\|_2}_{\text{Activation magnitude}} \cdot \underbrace{\|\mathbf{W}_l\|_F}_{\text{Weight importance}}. \tag{9}$$

We now proceed to examine the constituent components of the formulation in greater detail. Activation Jacobian $\mathbf{J}(\mathbf{h}_l)$ represents the local sensitivity of the activation function:

$$\mathbf{J}(\mathbf{h}_l) = \left.\frac{\partial \sigma(\mathbf{z}_l)}{\partial \mathbf{z}_l}\right|_{\mathbf{z}_l = \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l}. \tag{10}$$

For ReLU It takes the $\mathbf{J}(\mathbf{h}_l) = \text{diag}(\mathbb{I}[\mathbf{z}_l > 0])$ form. And the Frobenius norm $\|\cdot\|_F$ aggregates all partial derivatives.

Activation Norm $\|\mathbf{h}_l\|_2$ measures the Euclidean norm of post-activation outputs:

$$\|\mathbf{h}_l\|_2 = \sqrt{\sum_{i=1}^{n_l} (h_l^i)^2}, \tag{11}$$

and it also captures the overall signal strength through the layer.

Weight Matrix Norm $\|\mathbf{W}_l\|_F$ computes the Frobenius norm of the weight matrix:

$$\|\mathbf{W}_l\|_F = \sqrt{\sum_{i=1}^{n_l} \sum_{j=1}^{n_{l-1}} (w_{ij}^l)^2}, \tag{12}$$

and it also serves as a structural importance measure for the layer.

We now turn to the Computation Process in more detail. The evaluation proceeds through the five steps for each input $\mathbf{x}$:

1. Forward Pass:
$$\mathbf{z}_l = \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l. \tag{13}$$
2. Activation Computation:
$$\mathbf{h}_l = \sigma(\mathbf{z}_l). \tag{14}$$
3. Jacobian Evaluation:
$$\mathbf{J}(\mathbf{h}_l) = \begin{cases} \sigma'(\mathbf{z}_l) & \text{(element-wise)} \\ \mathbb{I}[\mathbf{z}_l > 0] & \text{(for ReLU).} \end{cases} \tag{15}$$
4. Norm Calculations:
$$\|\mathbf{J}(\mathbf{h}_l)\|_F = \sqrt{\sum_{i=1}^{n_l} (\sigma'(z_l^i))^2}, \tag{16}$$
$$\|\mathbf{h}_l\|_2 = \sqrt{\mathbf{h}_l^\top \mathbf{h}_l}, \tag{17}$$
$$\|\mathbf{W}_l\|_F = \sqrt{\text{tr}(\mathbf{W}_l^\top \mathbf{W}_l)}. \tag{18}$$
5. Layer Divergence:
$$\mathcal{D}_{\text{FC}}^{(l)} = \|\mathbf{J}(\mathbf{h}_l)\|_F \cdot \|\mathbf{h}_l\|_2 \cdot \|\mathbf{W}_l\|_F. \tag{19}$$

The product form captures three critical aspects of information flow:

$$\mathcal{D}_{\text{FC}}^{(l)} \propto \underbrace{\text{Sensitivity}}_{\mathbf{J}} \times \underbrace{\text{Signal Strength}}_{\mathbf{h}_l} \times \underbrace{\text{Parameter Significance}}_{\mathbf{W}_l} \tag{20}$$

Let us also highlight some important properties. Firstly, the scale invariant: $\mathcal{D}_{\text{FC}}^{(l)}(\alpha \mathbf{h}_l) = \mathcal{D}_{\text{FC}}^{(l)}(\mathbf{h}_l)$ for $\alpha > 0$. Secondly, the non-negativity: $\mathcal{D}_{\text{FC}}^{(l)} \geq 0$ with equality only for zero activations. And lastly, the composability. It states that total network divergence is the sum across layers:

$$\mathcal{D}_{\text{FC}}(\mathbf{x}) = \sum_{l=1}^{L} \mathcal{D}_{\text{FC}}^{(l)}(\mathbf{x}). \tag{21}$$

B.2 Divergence Explicit Representation for Convolutional Layers

Let us once again begin with the mathematical formulation. For convolutional layer l with input $\mathbf{X} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times C_{l-1}}$, the flow divergence is computed as:

$$\mathcal{D}_{\text{conv}}^{(l)}(\mathbf{X}) = \underbrace{\frac{1}{|\Omega_l|}}_{\text{Normalization}} \cdot \underbrace{\|\mathbf{A}_l\|_F}_{\text{Activation magnitude}} \cdot \underbrace{\|\mathbf{W}_l\|_F}_{\text{Weight significance}}, \tag{22}$$

where:

- $\Omega_l = H_l \times W_l \times C_l$ represents the *activation volume* with:
  - H_l, W_l: Spatial dimensions of output feature maps;
  - C_l: Number of output channels.
- $\mathbf{A}_l = \sigma(\mathbf{W}_l * \mathbf{X} + \mathbf{b}_l)$ denotes the *post-activation tensor* where:
  - $*$: Convolution operation with padding and stride;
  - σ: Element-wise activation function;
  - $\mathbf{W}_l \in \mathbb{R}^{k \times k \times C_{l-1} \times C_l}$: 4D convolution kernel;
  - $\mathbf{b}_l \in \mathbb{R}^{C_l}$: Bias vector.
- $\| \cdot \|_F$: Frobenius norm computing the *root-sum-square* of all elements.

We now proceed to the details of computational mechanics. The evaluation process involves Forward Pass Calculation in the form: $\mathbf{Z}_l = \mathbf{W}_l * \mathbf{X} + \mathbf{b}_l$ (pre-activation). It also includes the Activation Transformation: $\mathbf{A}_l = \phi(\mathbf{Z}_l)$ (where ϕ is ReLU, sigmoid, etc) and the Normalized Divergence Computation:

$$\mathcal{D}_{\text{conv}}^{(l)} = \frac{1}{|\Omega_l|} \sqrt{\sum_{i=1}^{H_l} \sum_{j=1}^{W_l} \sum_{k=1}^{C_l} |a_{ijk}|^2} \cdot \sqrt{\sum_{m=1}^{k} \sum_{n=1}^{k} \sum_{p=1}^{C_{l-1}} \sum_{q=1}^{C_l} |w_{mnpq}|^2}. \tag{23}$$

Additional characteristics and clarifications for the Convolutional Divergence Computation Parameters are provided in Table 2.

Table 2: Convolutional divergence computation parameters

Symbol	*Dimension*	*Interpretation*
k	Scalar	Convolution kernel size
$H_l \times W_l$	Spatial	Output feature map dimensions
C_l	Channels	Number of output filters
$\mathbf{W}_l$	$k \times k \times C_{l-1} \times C_l$	4D weight tensor
$\mathbf{A}_l$	$H_l \times W_l \times C_l$	3D activation tensor

The convolutional divergence measure possesses several important properties. It is scale-invariant, meaning that uniform scaling of activations and weights does not affect the value of the divergence, as expressed by

$$\mathcal{D}_{\text{conv}}^{(l)}(\alpha \mathbf{A}_l, \beta \mathbf{W}_l) = \mathcal{D}_{\text{conv}}^{(l)}(\mathbf{A}_l, \mathbf{W}_l) \quad \forall \alpha, \beta > 0. \tag{24}$$

The measure is also adaptable to architectural variations, automatically accounting for factors such as strided convolutions by adjusting output dimensions, dilated convolutions through the effective receptive field, and grouped convolutions via per-group computation. Furthermore, it is memory-efficient, as it requires only a single forward pass per layer to compute.

B.3 Divergence Explicit Representation for Self-Attention layers

We now consider the case of Single-Head Attention Divergence. For a basic self-attention mechanism, the divergence is computed as:

$$\mathcal{D}_{\text{attn}}^{\text{single}}(\mathbf{X}) = \frac{1}{n}\|\mathbf{A}\|_F \cdot \left(\|\mathbf{W}_Q\|_F + \|\mathbf{W}_K\|_F + \|\mathbf{W}_V\|_F\right), \tag{25}$$

where:

- $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$ is the input sequence matrix (n tokens, d_{model} dimensions);
- $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are learned projection matrices;
- $\mathbf{A} = \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V$ is the attention output;
- $\|\cdot\|_F$ denotes the Frobenius norm, measuring the "energy" of transformations;
- The $\frac{1}{n}$ term normalizes by sequence length.

We now examine the extension to Multi-Head Attention. The multi-head formulation generalizes this by considering H parallel attention heads:

$$\mathcal{D}_{\text{attn}}^{\text{multi}}(\mathbf{X}) = \sum_{h=1}^{H} \frac{1}{n}\|\mathbf{A}^h\|_F \cdot \left(\|\mathbf{W}_Q^h\|_F + \|\mathbf{W}_K^h\|_F + \|\mathbf{W}_V^h\|_F\right). \tag{26}$$

It is worth separately noting a few additional remarks. Firstly, each head h has independent projections $\mathbf{W}_Q^h, \mathbf{W}_K^h \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_V^h \in \mathbb{R}^{d_{\text{model}} \times d_v}$. Secondly,

$$\mathbf{A}^h = \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q^h(\mathbf{X}\mathbf{W}_K^h)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V^h \tag{27}$$

represents head-specific attention. Lastly, the sum over heads captures total information transformation.

We consider the four steps of the Derivation Process:

1. Single-Head Basis. Start with the basic attention divergence:
$$\mathcal{D}_{\text{attn}}^{\text{base}} = \frac{\|\text{Attention}(\mathbf{X})\|_F}{n} \cdot \|\theta\|_F, \tag{28}$$
where θ contains all projection parameters.
2. Parameter Decomposition. Separate the Frobenius norms by projection type:
$$\|\theta\|_F \to \|\mathbf{W}_Q\|_F + \|\mathbf{W}_K\|_F + \|\mathbf{W}_V\|_F. \tag{29}$$
3. Multi-Head Expansion. In the case of H heads, the measure becomes additive, as each head operates on an independent subspace, the concatenated output preserves dimensional scaling, and the $\frac{1}{n}$ normalization remains valid for each head individually.
4. Residual Consideration. In practice, we account for
$$\mathcal{D}_{\text{attn}}^{\text{final}} = \mathcal{D}_{\text{attn}}^{\text{multi}} + \lambda\|\mathbf{W}_O\|_F, \tag{30}$$
where $\mathbf{W}_O$ is the output projection and λ balances terms.

The multi-head divergence measure has three key aspects:

1. Attention Pattern Term ($\|\mathbf{A}^h\|_F$) measures how strongly inputs are transformed by the attention weights.

2. Projection Importance Term ($\sum \|\mathbf{W}_*^h\|_F$) captures the magnitude of learned query/key/value transformations.
3. Normalization Factor ($\frac{1}{n}$) ensures comparability across varying sequence lengths.

The following theorem serves as the theoretical justification for the formulation presented above.

Theorem 4 (Additive Composition)**.** *For independent attention heads, the total divergence equals the sum of head-specific divergences:*

$$\mathcal{D}_{attn}^{multi}(\mathbf{X}) = \sum_{h=1}^{H} \mathcal{D}_{attn}^{h}(\mathbf{X}). \tag{31}$$

Proof. Follows from the linearity of the Frobenius norm and the block-diagonal structure of multi-head projections. □

C Divergence Computation for Different Layer Types

C.1 Divergence Evaluation Algorithm for Fully Connected Architectures

Let us consider the algorithms for calculating divergence using the above layer types as an example. Firstly, let us take a look at fully connected networks. The information flow can be quantified using Algorithm 2, which tracks how signal transformations evolve across successive layers.

Algorithm 2 Measuring Divergence of Information Flow in FC Networks

Require: Input vector $\mathbf{x}$, weight matrices $\{\mathbf{W}_l\}$, biases $\{\mathbf{b}_l\}$
Ensure: Total information divergence $\mathcal{D}_{\text{FC}}$
Initialize divergence accumulator: $\mathcal{D}_{\text{FC}} \leftarrow 0$
Set initial activation: $\mathbf{h}_0 \leftarrow \mathbf{x}$
for each layer $l = 1$ to L **do**
    Compute pre-activation: $\mathbf{z}_l \leftarrow \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l$
    Apply nonlinearity: $\mathbf{h}_l \leftarrow \sigma(\mathbf{z}_l)$
    Measure layer transformation: $\delta_l \leftarrow \|\mathbf{h}_l\|_2 \cdot \|\mathbf{W}_l\|_F$
    Accumulate divergence: $\mathcal{D}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} + \delta_l$
end for
return $\mathcal{D}_{\text{FC}}$

From a computational perspective, the time complexity is dominated by matrix-vector products and scales as $O\left(\sum_{l=1}^{L} n_l n_{l-1}\right)$, while the space complexity is determined by the need to store layer activations, requiring $O\left(\sum_{l=1}^{L} n_l\right)$ memory.

It also should be mentioned that ReLU activations simplify the divergence measure to:

$$\delta_l^{\text{ReLU}} = \|\max(0, \mathbf{z}_l)\|_2 \cdot \|\mathbf{W}_l\|_F, \tag{32}$$

while the Frobenius norm $\|\mathbf{W}_l\|_F$ serves as an automatic importance weighting for each layer's contribution.

C.2 Divergence Evaluation Algorithm for Convolutional Architectures

For convolutional networks, Algorithm 3 measures how spatial feature representations transform across the network depth.

Algorithm 3 Measuring Divergence of Information Flow in Convolutional Networks

Require: Input tensor $\mathbf{X}$, convolution kernels $\{\mathbf{W}_l\}$, biases $\{\mathbf{b}_l\}$
Ensure: Total spatial divergence $\mathcal{D}_{\text{conv}}$
Initialize divergence measure: $\mathcal{D}_{\text{conv}} \leftarrow 0$
Set input features: $\mathbf{A}_0 \leftarrow \mathbf{X}$
for each conv layer $l = 1$ to L **do**
    Compute convolution: $\mathbf{Z}_l \leftarrow \mathbf{W}_l * \mathbf{A}_{l-1} + \mathbf{b}_l$
    Apply activation: $\mathbf{A}_l \leftarrow \sigma(\mathbf{Z}_l)$
    Get tensor dimensions: $(H_l, W_l, C_l) \leftarrow \text{shape}(\mathbf{A}_l)$
    Compute normalized divergence: $\delta_l \leftarrow \frac{\|\mathbf{A}_l\|_F \cdot \|\mathbf{W}_l\|_F}{H_l W_l C_l}$
    Update total: $\mathcal{D}_{\text{conv}} \leftarrow \mathcal{D}_{\text{conv}} + \delta_l$
end for
return $\mathcal{D}_{\text{conv}}$

The complexity analysis reveals that the time complexity for $k \times k$ convolutions is $O\left(\sum_{l=1}^{L} H_l W_l C_l C_{l-1} k^2\right)$, while the memory requirements for storing feature maps amount to $O\left(\sum_{l=1}^{L} H_l W_l C_l\right)$.

Implementation-wise, strided operations require appropriate dimension adjustments, while batch normalization layers can be seamlessly integrated by modifying the pre-activation computation. Pooling layers, although part of the computational path, contribute zero parameter divergence.

C.3 Divergence Evaluation Algorithm for Attention-Based Architectures

Self-attention mechanisms require specialized flow measurement as detailed in Algorithm 4, capturing both feature transformation and attention pattern evolution.

Algorithm 4 Measuring Divergence of Information Flow in Attention-Based Networks

Require: Input sequence $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$, projection weights $\{\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h\}$
Ensure: Total attention divergence $\mathcal{D}_{\text{attn}}$
1: Initialize divergence: $\mathcal{D}_{\text{attn}} \leftarrow 0$
2: **for** each head $h = 1$ to H **do**
3: Project queries: $\mathbf{Q}^h \leftarrow \mathbf{X}\mathbf{W}_Q^h$
4: Project keys: $\mathbf{K}^h \leftarrow \mathbf{X}\mathbf{W}_K^h$
5: Project values: $\mathbf{V}^h \leftarrow \mathbf{X}\mathbf{W}_V^h$
6: Compute attention: $\mathbf{S}^h \leftarrow \text{softmax}(\mathbf{Q}^h(\mathbf{K}^h)^\top/\sqrt{d_k})$
7: Transform features: $\mathbf{O}^h \leftarrow \mathbf{S}^h\mathbf{V}^h$
8: Measure head divergence: $\delta_h \leftarrow \frac{\|\mathbf{A}^h\|_F}{n} \cdot \sum_{P\in\{Q,K,V\}} \|\mathbf{W}_P^h\|_F$
9: Accumulate: $\mathcal{D}_{\text{attn}} \leftarrow \mathcal{D}_{\text{attn}} + \delta_h$
10: **end for**
11: **return** $\mathcal{D}_{\text{attn}}$

The computational requirements for the attention mechanism include a time complexity of $O(Hn^2d_k + Hnd_v^2)$, which accounts for both attention score computation and value transformations, and a space complexity of $O(Hnd_v)$ for storing the attention outputs.

The analysis reveals that multi-head processing requires per-head divergence computation, while layer normalization and residual connections affect information flow and must be handled accordingly. The measure captures both attention dynamics and value transformations, with total transformer block divergence decomposing into attention and feed-forward components:

$$\mathcal{D}_{\text{block}} = \mathcal{D}_{\text{attn}} + \mathcal{D}_{\text{ffn}}. \tag{33}$$

D ITERATIVE DIVERGENCE-AWARE PRUNING ALGORITHM

Algorithm 5 Iterative Divergence-Aware Pruning (IDAP)

$\mathcal{M}_0$: Initial trained model
$\mathcal{V}$: Validation dataset
τ: Maximum allowable performance degradation
K: Number of pruning iterations
ρ_0: Base pruning ratio
α: Aggressiveness coefficient
$\mathcal{M}^*$: Optimally pruned model
$\mathcal{W}^*$: Final weight configuration

Initialize:
$\mathcal{D} \leftarrow \text{ComputeDivergence}(\mathcal{M}_0)$ ▷ Sec. C.1-C.3
$\mathbf{w} \leftarrow \text{SortWeights}(\mathcal{M}_0.\text{params}, \mathcal{D})$
$\mathcal{P} \leftarrow \{\}$ ▷ Pruning history archive
for $k \leftarrow 1\ K$ **do**
    Determine current pruning ratio:

$$\rho_k \leftarrow \rho_0 \cdot (1 + k/K)^\alpha$$

    Compute divergence threshold:

$$\theta_k \leftarrow \text{Quantile}(\mathbf{w}, \rho_k)$$

    Generate pruning mask:

$$\mathbf{m}_k \leftarrow \mathbb{I}[\mathcal{D} > \theta_k]$$

    Evaluate pruned model:

$$\text{Perf}_k \leftarrow \text{Evaluate}(\mathcal{M}_0 \odot \mathbf{m}_k, \mathcal{V})$$

    if $\text{Perf}_0 - \text{Perf}_k > \tau$ **then**
        Revert to $\mathbf{m}_{k-1}$
        exit loop
    else
        $\mathcal{P} \leftarrow \mathcal{P} \cup (\rho_k, \text{Perf}_k)$
    end if
end for
Select optimal configuration:

$$\rho^* \leftarrow \max\{\rho \in \mathcal{P} \mid \text{Perf}_0 - \text{Perf}(\rho) \leq \tau\}$$

Apply final mask:

$$\mathcal{M}^* \leftarrow \text{FineTune}(\mathcal{M}_0 \odot \mathbf{m}^*)$$

return $\mathcal{M}^*, \mathcal{W}^*$

E LAYER REMOVAL BASED ON INFORMATION FLOW DIVERGENCE ANALYSIS

Algorithm 6 Layer Removal Based on Information Flow Divergence Analysis

Require:
- Pruned network $\mathcal{N}'$ from Stage I
- Validation set $\mathcal{D}_{\text{val}}$
- Target error reduction ratio γ
- Maximum layer removal budget $R_{\max}$

Ensure:
- Optimally compressed network $\mathcal{N}^*$
- Set of removed layers $\mathcal{L}_{\text{removed}}$

Initialize removal candidate set: $\mathcal{L}_{\text{candidates}} \leftarrow \text{SortLayersByFlow}(\mathcal{N}')$
Initialize error reduction tracker: $\Delta E \leftarrow 0$
Initialize removal counter: $r \leftarrow 0$
while $r < R_{\max}$ **and** $\Delta E < \gamma$ **do**
  Select layer with minimal flow: $l^* \leftarrow \arg\min_{l \in \mathcal{L}_{\text{candidates}}} \mathcal{D}_l$
  Perform Layer Replacement:
  Create temporary network: $\mathcal{N}_{\text{temp}} \leftarrow \mathcal{N}'$
  Replace l^* with identity mapping: $\mathcal{N}_{\text{temp}}.l^* \leftarrow \text{Identity*}()$
  Fine-tune replacement: $\mathcal{N}_{\text{temp}} \leftarrow \text{FineTune}(\mathcal{N}_{\text{temp}}, \mathcal{D}_{\text{val}})$
  Evaluate Impact:
  Compute error reduction: $\delta E \leftarrow E(\mathcal{N}') - E(\mathcal{N}_{\text{temp}})$
  if $\delta E > 0$ **then**
    Accept removal: $\mathcal{N}' \leftarrow \mathcal{N}_{\text{temp}}$
    Update candidates: $\mathcal{L}_{\text{candidates}} \leftarrow \mathcal{L}_{\text{candidates}} \setminus \{l^*\}$
    Record removal: $\mathcal{L}_{\text{removed}} \leftarrow \mathcal{L}_{\text{removed}} \cup \{l^*\}$
    Update metrics: $\Delta E \leftarrow \Delta E + \delta E$, $r \leftarrow r + 1$
  else
    Mark layer as essential: $\mathcal{L}_{\text{candidates}} \leftarrow \mathcal{L}_{\text{candidates}} \setminus \{l^*\}$
  end if
end while
return $\mathcal{N}^* \leftarrow \textbf{FinalFineTune}(\mathcal{N}'), \mathcal{L}_{\text{removed}}$

F DETAILED RESULTS

Table 3: Model compression dynamics of ResNet-50 on CIFAR-10 using the two-stage IDAP++ framework

Pruning Step	Stage	Params (M)	GFlops	Top-1 Acc. (%)	Top-5 Acc. (%)	Δ Top-1 Acc.
1	Baseline	23.53	4.09	98.20	99.86	0.00
2	Filter Prune	22.27	3.89	97.66	99.85	-0.54
3	Filter Prune	21.20	3.66	97.23	99.84	-0.97
4	Filter Prune	19.89	3.46	96.99	99.73	-1.21
5	Filter Prune	18.78	3.31	97.11	99.89	-1.09
6	Filter Prune	17.54	3.13	97.74	99.89	-0.46
7	Filter Prune	16.45	2.90	97.62	99.84	-0.58
8	Filter Prune	15.50	2.73	97.93	99.87	-0.27
9	Filter Prune	14.62	2.61	98.09	99.76	-0.11
10	Filter Prune	14.14	2.52	98.05	99.75	-0.15
11	Filter Prune	13.50	2.37	97.87	99.77	-0.33
12	Filter Prune	12.98	2.26	97.85	99.81	-0.35
13	Filter Prune	12.37	2.15	97.84	99.77	-0.36
14	Filter Prune	11.82	2.08	97.77	99.79	-0.43
15	Filter Prune	11.26	1.98	97.70	99.76	-0.50
16	Filter Prune	11.02	1.94	97.85	99.80	-0.35
17	Filter Prune	10.77	1.89	97.56	99.81	-0.64
18	Filter Prune	10.53	1.85	97.50	99.79	-0.70
19	Filter Prune	10.28	1.81	97.42	99.80	-0.78
20	Filter Prune	10.04	1.77	97.35	99.78	-0.85
21	Filter Prune	9.79	1.73	97.28	99.75	-0.92
22	Filter Prune	9.55	1.68	97.50	99.77	-0.70
23	Filter Prune	9.30	1.49	97.52	99.78	-0.68
24	Filter Prune	9.05	1.45	97.08	99.77	-1.12
25	Filter Prune	8.81	1.40	97.50	99.80	-0.70
26	Filter Prune	8.56	1.34	97.40	99.81	-0.80
27	Filter Prune	8.32	1.30	96.91	99.79	-1.29
28	Filter Prune	8.07	1.26	97.25	99.78	-0.95
29	Filter Prune	7.83	1.22	97.52	99.80	-0.68
30	Filter Prune	7.57	1.19	97.63	99.81	-0.57
31	Layer Trunc	6.73	1.17	97.22	99.39	-0.98
32	Layer Trunc	6.67	1.16	96.78	98.94	-1.42
33	Layer Trunc	6.62	1.15	96.42	98.57	-1.78
34	Layer Trunc	6.56	1.14	95.57	98.03	-2.63
35	Final Fine-Tune	6.56	1.14	95.98	98.12	-2.22

Table 4: Inference time summary by architecture (RTX 3060, batch size = 1, FP32)

Architecture	Inference Time		Speedup
	Base (ms)	Pruned (ms)	x
ResNet-50	8.5	4.3	1.98
EfficientNet-B4	8.8	4.6	1.91
ViT-Base/16	33.2	20.3	1.64
MobileNetV3-L	4.1	1.9	2.16
DenseNet-121	6.2	3.3	1.88
ConvNeXt-Small	17.5	10.5	1.67
VGG19-BN	38.2	18.0	2.12
ShuffleNetV2 x2.0	3.5	1.8	1.94
Faster R-CNN (ResNet-50)	48.0	28.0	1.71
YOLOv4 (ShuffleNetV2)	12.5	6.8	1.84
DETR (ViT-Base/16)	75.0	48.0	1.56
FCN (VGG19-BN)	52.0	26.5	1.96
U-Net (ResNet-50)	28.0	15.5	1.81
SegFormer (ViT-Base/16)	65.0	41.0	1.59
BERT Base	45.0	28.0	1.61
GPT-2 Base	120.0	80.0	1.50
T5 Base	95.0	62.0	1.53