IDAP++: Advancing Divergence-Aware Pruning with Joint Filter and Layer Optimization

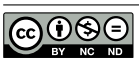
ALEKSEI SAMARIN, Wayy LLC, USA and ITMO University, Russian Federation
ARTEM NAZARENKO, Wayy LLC, USA and ITMO University, Russian Federation
EGOR KOTENKO, Wayy LLC, USA and Saint Petersburg State University, Russian Federation
ALEXANDER SAVELEV, Wayy LLC, USA and ITMO University, Russian Federation
ALEKSEI TOROPOV, Wayy LLC, USA and ITMO University, Russian Federation
ALEXANDR MOTYKO, Wayy LLC, USA
VALENTIN MALYKH, Wayy LLC, USA and ITMO University, Russian Federation

Modern knowledge and large volumes of data are increasingly encoded within neural networks, making the task of simplifying their structures and reducing the number of parameters especially relevant, both to improve efficiency and to facilitate deployment in resource-constrained environments. 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 upon the concept of tensor flow divergence, which quantifies how information transforms 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 state-of-the-art solutions and outperform 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.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Factorization methods**.

Additional Key Words and Phrases: Neural Network Pruning, Information Flow Divergence, Model Compression, Architecture Optimization

Authors' Contact Information: Aleksei Samarin, Wayy LLC, Miami, USA and ITMO University, St. Petersburg, Russian Federation, avsamarin@itmo.ru; Artem Nazarenko, Wayy LLC, Miami, USA and ITMO University, St. Petersburg, Russian Federation, aanazarenko@itmo.ru; Egor Kotenko, Wayy LLC, Miami, USA and Saint Petersburg State University, St. Petersburg, Russian Federation, kotenkoed@gmail.com; Alexander Savelev, Wayy LLC, Miami, USA and ITMO University, St. Petersburg, Russian Federation, algsavelev@gmail.com; Aleksei Toropov, Wayy LLC, Miami, USA and ITMO University, St. Petersburg, Russian Federation, toropov.ag@hotmail.com; Alexandr Motyko, Wayy LLC, Miami, USA, motyko.alexandr@yandex.ru; Valentin Malykh, Wayy LLC, Miami, USA and ITMO University, St. Petersburg, Russian Federation, valentin.malykh@phystech.edu.



ACM 2836-6573/2026/2-ART45
https://doi.org/10.1145/3786659

ACM Reference Format:
Aleksei Samarin, Artem Nazarenko, Egor Kotenko, Alexander Savelev, Aleksei Toropov, Alexandr Motyko, and Valentin Malykh. 2026. IDAP++: Advancing Divergence-Aware Pruning with Joint Filter and Layer Optimization. *Proc. ACM Manag. Data* 4, 1, Article 45 (February 2026), 28 pages. https://doi.org/10.1145/3786659

1 Introduction

Modern artificial intelligence (AI) systems are rapidly penetrating all areas of human activity, transforming key industries and high-tech products [6, 43, 74, 75, 114, 127]. Today, AI is used in mobile devices [40, 65], autonomous vehicles [12, 47], healthcare [10, 137], finance [39, 94], industry [42, 103], and scientific research [76, 124]. Most of these achievements are based on deep neural networks (DNNs) [109, 118], which over the past decade have demonstrated revolutionary results across numerous tasks, from computer vision [82, 91, 138] and natural language processing [41, 81, 112] to generative models [62, 101, 132] and control systems [78, 96, 120]. Prominent examples include large language models such as GPT-4 [88] and Gemini [113], medical diagnostic systems based on convolutional neural networks [16], as well as image and video generation models like DALL·E [71] and Stable Diffusion [18, 35, 90]. Deep learning advancements have enabled unprecedented accuracy, adaptability, and quality in intelligent systems.

Another crucial role of neural networks in the modern world deserves special attention — they increasingly serve as repositories of a significant portion of contemporary knowledge and large-scale data. Indeed, more and more information is now encoded within various neural architectures, particularly in large language models (LLMs) and advanced multimodal systems. These models are not limited to processing tasks; rather, they act as complex storage mechanisms for structured, high-dimensional representations learned from vast and diverse training datasets. As the use of such models expands across domains and their architectures grow in size and complexity, the amount of embedded knowledge they contain continues to increase. As a result, neural networks are becoming essential not only for computation but also for knowledge storage, representation, and interpretation.

The growing integration of neural networks into data management systems creates new challenges at the intersection of machine learning and database technologies. Modern database systems increasingly leverage embedded models for query optimization [20, 72, 73], approximate query processing [34, 56, 61], and in-database analytics [70]. As models become larger and more complex, their computational and storage demands directly impact database performance and resource utilization. Efficient model compression is therefore not merely a machine learning concern but a critical database systems challenge, enabling practical deployment of AI capabilities within resource-constrained database environments while maintaining the low-latency requirements essential for interactive data systems.

However, all the impressive achievements have come at the cost of an exponential increase in the scale of neural network models [3]. Modern architectures now contain hundreds of millions, and in some cases, hundreds of billions of parameters, making them extremely resource-intensive both during training and in real-world deployment [55]. For example, language models like GPT-4 or LLaMA 3 [27] require vast computational clusters composed of specialized accelerators [49] to function effectively. The associated costs include not only the necessary time and energy for training but also significant operational expenses related to their deployment [2] — from high electricity consumption in data centers to technical limitations when integrating such models into mobile devices [9] or embedded devices [87].

Given these growing demands for resources and the associated costs, optimizing neural network models has become a critical challenge for the AI research and engineering community [45, 52, 98]. Reducing computational requirements without sacrificing model quality is essential to ensure

accessibility, ecological sustainability, and large-scale practical deployment of AI technologies [57, 83, 86, 104, 123, 128].

Numerous optimization strategies have been proposed in this context, including quantization [26, 60, 64, 130], weight factorization [14, 29, 37, 95], low-bitwidth representations [17, 106, 126], and hardware accelerators [7, 92, 119]. However, most of these approaches face limitations in terms of universality, implementation complexity, or inevitable trade-offs in model accuracy. Among the most promising optimization directions is pruning [13, 24, 25, 33, 59, 108, 136] — a technique for structurally simplifying neural networks by removing the least significant parameters or neurons. Pruning is not merely an engineering method for reducing computations, but a fundamental tool for understanding and refining neural network architectures. The core idea involves identifying and eliminating redundant components while preserving the model's key functional properties. In practice, pruning has been successfully applied to a wide range of tasks, including image classification [1, 84, 111], text processing [53, 68, 102], and generative models [5, 44, 100], achieving substantial reductions in model size and computational cost without noticeable performance degradation.

The practical benefits of pruning include lower memory and hardware requirements, faster inference, simplified deployment on mobile and embedded platforms, and improved energy efficiency. However, existing pruning techniques still exhibit significant drawbacks: many rely on heuristics, scale poorly to large architectures, or insufficiently account for the complex dynamics of information propagation within neural networks [1, 5, 13, 24, 25, 33, 44, 53, 59, 68, 84, 100, 102, 108, 111, 136].

This work introduces a fundamentally new approach to neural network compression by addressing redundancy at every level of architecture through a theoretically grounded, two-stage optimization framework. At its core lies the concept of information flow divergence – a rigorously defined metric that quantifies how signals evolve as they propagate through the network's layers. By treating neural networks as dynamic systems where each transformation carries measurable informational value, we develop a compression methodology that operates holistically across both the width and depth of architectures.

The first stage of our method focuses on filter-level optimization, where divergence [19, 48, 66, 69, 89, 93, 117] measurements identify and prune redundant parameters while preserving critical information pathways [28, 79, 99, 105, 129, 135]. This process goes beyond conventional pruning by not just removing weak connections but actively reshaping the network's capacity to maintain efficient signal flow. Building upon this foundation, the second stage introduces architectural compression through selective layer removal, where entire computational blocks are evaluated and consolidated based on their contribution to the overall information throughput.

What makes this two-stage approach unique is its comprehensive nature, where traditional methods might focus on parameter count or layer count reduction, our framework simultaneously optimizes both dimensions while respecting the underlying information dynamics. The mathematical formalism of flow divergence provides a unifying language for these operations, connecting practical compression decisions to fundamental principles of signal propagation.

We support this methodology with complete algorithmic specifications, including efficient computation schemes for various layer types, and demonstrate through extensive experiments that such holistic compression yields superior results compared to isolated optimization strategies. The framework's effectiveness is shown across diverse architectures, from convolutional networks to transformers, consistently achieving substantial model size reduction without compromising the network's ability to process and transform information effectively.

This work ultimately provides more than just a compression tool; it offers a new perspective on neural network design where architectural decisions are guided by measurable information flow

characteristics, enabling the creation of models that are not just smaller but also more efficient in their computational structure.

The compression methodology presented in this work addresses several pressing needs in modern data management:

- **In-Database Model Serving:** As enterprises deploy ML models directly within database systems for real-time inference model size directly affects memory footprint and query latency. Our approach enables larger models to operate within the strict memory constraints of database servers while maintaining inference speeds suitable for interactive query processing.
- **Embedded Vector Representations:** Modern databases increasingly store neural embeddings for similarity search and recommendation systems. Compressed models reduce the storage overhead for these embedded representations while maintaining their semantic fidelity, enabling more efficient vector operations directly within the database engine.
- **Edge Database Systems:** For distributed and edge database deployments where computational resources are limited, our compression technique facilitates the deployment of sophisticated models that would otherwise be impractical, bringing advanced analytics closer to data sources.
- **Energy-Efficient Data Centers:** With the growing carbon footprint of data centers, reducing the computational requirements of ML components within database systems contributes to more sustainable data management practices, aligning with green computing initiatives in large-scale data infrastructure.

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**:
  - ~75-90% CNN pruning with <2% accuracy drop.
  - >70% transformers pruning while maintaining ~98%+ baseline accuracy.
  - >40% faster inference post-compression.

2 Problem Statement

Modern neural networks are often overparameterized, with many operations adding no real benefit to performance, resulting in unnecessary complexity [77].

The core challenge in optimizing neural networks lies in developing methods that simplify architectures and reduce computational complexity while preserving the ability to generate accurate and reliable results, whether for classification, text generation, image synthesis, or other tasks. It is essential to maintain not only the target performance level but also the model's robustness, generalization capacity, and adaptability to real-world conditions.

This challenge is difficult due to architectural heterogeneity, complex internal dynamics, and limited interpretability of pruning effects. Scaling such methods to large models also requires highly efficient optimization.

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 instantaneous information flow is captured by the differential change:

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

Here, $\phi(s)$ represents the rate of feature transformation at depth s, capturing how rapidly the network modifies its internal representations. This continuous formulation allows us to model both discrete and continuous architectures within a unified framework, treating the network as a dynamical system that evolves input data through a sequence of transformations. The derivative-based approach naturally handles residual connections and provides a mathematical foundation for analyzing information propagation independent of specific architectural choices.

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:

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

For practical computation in discrete networks with L layers:

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

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.

3.1.3 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:

$$\hat{\mathcal{D}}_l = \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_l(\mathbf{x}_i) \cdot \left(1 + \frac{\text{Var}(\mathbf{T}_l)}{\sigma^2_{\max}}\right)^{-1}. \tag{5}$$

where:

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

3.1.4 Key Properties of the introduced divergence measure. The divergence measure satisfies two fundamental properties, which are formulated as corresponding lemmas.

Lemma 3.1 (Scale Invariance). *For any $\alpha > 0$:*

$$\mathcal{D}_l(\alpha\mathbf{T}_l, \alpha\mathbf{T}_{l+1}) = \mathcal{D}_l(\mathbf{T}_l, \mathbf{T}_{l+1}). \tag{6}$$

Lemma 3.2 (Additive Composition). *For sequential transformations:*

$$\mathcal{D}_{l\to l+2} = \mathcal{D}_l \cdot \mathcal{D}_{l+1} + \mathcal{O}(\|\Delta\mathbf{T}\|^3). \tag{7}$$

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 which we perform a reduction of insignificant filters, and at the second - 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).

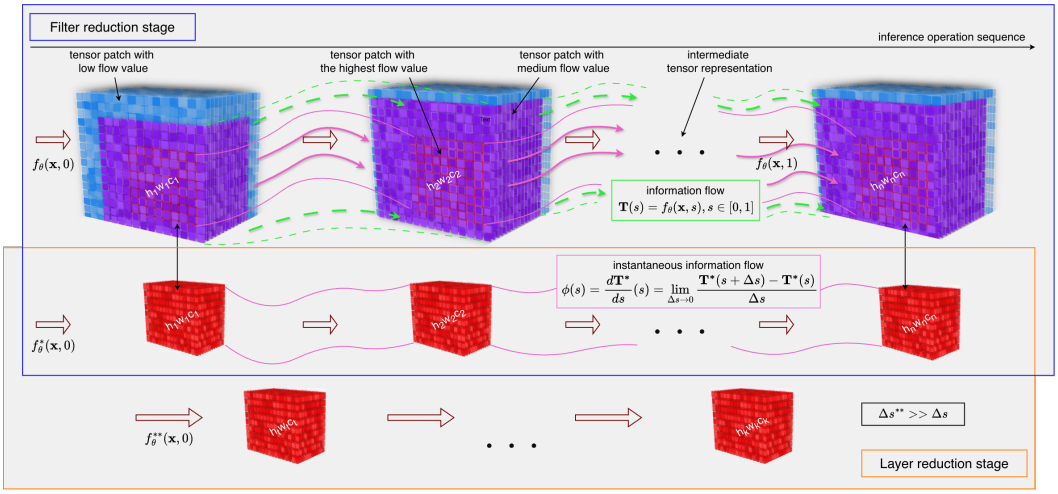


Fig. 1. Visualization of information flow through network depth. Arrows represent derivative-based flow measurements at different depth coordinates s.

3.2 Stage 1: Filters Reduction (IDAP algorithm)

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, let's formalize the concept of divergence for the most basic types of main layers of a neural network.

3.2.1 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:

$$\mathcal{D}_{\text{FC}}^{(l)}(\mathbf{x}) = \underbrace{\|\mathbf{J}(\mathbf{h}_l)\|_F}_{\text{Activation sensitivity}} \cdot \underbrace{\|\mathbf{h}_l\|_2}_{\text{Activation magnitude}} \cdot \underbrace{\|\mathbf{W}_l\|_F}_{\text{Weight importance}}. \tag{8}$$

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:

$$\mathbf{J}(\mathbf{h}_l) = \left.\frac{\partial \sigma(\mathbf{z}_l)}{\partial \mathbf{z}_l}\right|_{\mathbf{z}_l = \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l}. \tag{9}$$

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:

$$\|\mathbf{h}_l\|_2 = \sqrt{\sum_{i=1}^{n_l} (h_l^i)^2}, \tag{10}$$

and it also captures the overall signal strength through the layer. This Euclidean norm serves as a proxy for signal strength flowing through the layer, quantifying the overall activation energy across all neurons. Unlike simple magnitude-based pruning criteria, this norm captures collective behavior across the entire layer, ensuring that pruning decisions consider global information flow rather than individual neuron activity. The squared formulation emphasizes larger activations while maintaining differentiability, crucial for our gradient-based flow analysis.

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

$$\|\mathbf{W}_l\|_F = \sqrt{\sum_{i=1}^{n_l} \sum_{j=1}^{n_{l-1}} (w_{ij}^l)^2}, \tag{11}$$

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:

$$\mathbf{z}_l = \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l. \tag{12}$$

(2) Activation Computation:

$$\mathbf{h}_l = \sigma(\mathbf{z}_l). \tag{13}$$

(3) Jacobian Evaluation:

$$\mathbf{J}(\mathbf{h}_l) = \begin{cases} \sigma'(\mathbf{z}_l) & \text{(element-wise)} \\ \mathbb{I}[\mathbf{z}_l > 0] & \text{(for ReLU).} \end{cases} \tag{14}$$

(4) Norm Calculations:

$$\|\mathbf{J}(\mathbf{h}_l)\|_F = \sqrt{\sum_{i=1}^{n_l} (\sigma'(z_l^i))^2}, \|\mathbf{h}_l\|_2 = \sqrt{\mathbf{h}_l^\top \mathbf{h}_l}, \|\mathbf{W}_l\|_F = \sqrt{\mathrm{tr}(\mathbf{W}_l^\top \mathbf{W}_l)}. \tag{15}$$

(5) Layer Divergence:

$$\mathcal{D}_{\mathrm{FC}}^{(l)} = \|\mathbf{J}(\mathbf{h}_l)\|_F \cdot \|\mathbf{h}_l\|_2 \cdot \|\mathbf{W}_l\|_F. \tag{16}$$

The product form captures three critical aspects of information flow:

$$\mathcal{D}_{\mathrm{FC}}^{(l)} \propto \underbrace{\text{Sensitivity}}_{\mathbf{J}} \times \underbrace{\text{Signal Strength}}_{\mathbf{h}_l} \times \underbrace{\text{Parameter Significance}}_{\mathbf{W}_l}. \tag{17}$$

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

$$\mathcal{D}_{\mathrm{FC}}(\mathbf{x}) = \sum_{l=1}^{L} \mathcal{D}_{\mathrm{FC}}^{(l)}(\mathbf{x}). \tag{18}$$

3.2.2 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:

$$\mathcal{D}_{\mathrm{conv}}^{(l)}(\mathbf{X}) = \underbrace{\frac{1}{|\Omega_l|}}_{\text{Normalization}} \cdot \underbrace{\|\mathbf{A}_l\|_F}_{\text{Activation magnitude}} \cdot \underbrace{\|\mathbf{W}_l\|_F}_{\text{Weight significance}}, \tag{19}$$

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:

$$\mathcal{D}_{\mathrm{conv}}^{(l)} = \frac{1}{|\Omega_l|} \sqrt{\sum_{i=1}^{H_l} \sum_{j=1}^{W_l} \sum_{k=1}^{C_l} |a_{ijk}|^2} \cdot \sqrt{\sum_{m=1}^{k} \sum_{n=1}^{k} \sum_{p=1}^{C_{l-1}} \sum_{q=1}^{C_l} |w_{mnpq}|^2}. \tag{20}$$

Additional characteristics and clarifications for the Convolutional Divergence Computation Parameters are provided in Table 1.

Table 1. 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

$$\mathcal{D}_{\text{conv}}^{(l)}(\alpha \mathbf{A}_l, \beta \mathbf{W}_l) = \mathcal{D}_{\text{conv}}^{(l)}(\mathbf{A}_l, \mathbf{W}_l) \quad \forall \alpha, \beta > 0. \tag{21}$$

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.

3.2.3 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:

$$\mathcal{D}_{\text{attn}}^{\text{single}}(\mathbf{X}) = \frac{1}{n}\|\mathbf{A}\|_F \cdot \left(\|\mathbf{W}_Q\|_F + \|\mathbf{W}_K\|_F + \|\mathbf{W}_V\|_F\right), \tag{22}$$

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;
- $$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V \tag{23}$$
  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:

$$\mathcal{D}_{\text{attn}}^{\text{multi}}(\mathbf{X}) = \sum_{h=1}^{H} \frac{1}{n}\|\mathbf{A}^h\|_F \cdot \left(\|\mathbf{W}_Q^h\|_F + \|\mathbf{W}_K^h\|_F + \|\mathbf{W}_V^h\|_F\right). \tag{24}$$

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,

$$\mathbf{A}^h = \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q^h(\mathbf{X}\mathbf{W}_K^h)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V^h \tag{25}$$

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:

$$\mathcal{D}_{\text{attn}}^{\text{base}} = \frac{\|\text{Attention}(\mathbf{X})\|_F}{n} \cdot \|\theta\|_F, \tag{26}$$

where θ contains all projection parameters.

(2) Parameter Decomposition. Separate the Frobenius norms by projection type:

$$\|\theta\|_F \to \|\mathbf{W}_Q\|_F + \|\mathbf{W}_K\|_F + \|\mathbf{W}_V\|_F. \tag{27}$$

(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

$$\mathcal{D}_{\text{attn}}^{\text{final}} = \mathcal{D}_{\text{attn}}^{\text{multi}} + \lambda\|\mathbf{W}_O\|_F, \tag{28}$$

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

$$\mathcal{D}_{attn}^{multi}(\mathbf{X}) = \sum_{h=1}^{H} \mathcal{D}_{attn}^{h}(\mathbf{X}). \tag{29}$$

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

The application of our divergence measure to self-attention layers requires careful consideration of their unique structural properties. Unlike convolutional or fully-connected layers that perform local transformations, self-attention operates through global interactions across the sequence dimension. Our formulation in Equations (24)-(26) captures this by decomposing the attention mechanism into its fundamental components: query-key similarity computation and value aggregation. The Frobenius norm of the attention matrix $\|\mathbf{A}\|_F$ measures the overall strength of attention patterns, while the projection norms $\|\mathbf{W}_Q\|_F$, $\|\mathbf{W}_K\|_F$, and $\|\mathbf{W}_V\|_F$ assess the importance of each transformation subspace.

For multi-head attention, the additive composition in Equation (25) reflects the independent contribution of each head to the overall information flow. This design allows our method to identify and prune specific heads that contribute minimally to the network's functionality, enabling more granular compression than whole-layer removal. The normalization factor $\frac{1}{n}$ ensures consistent measurement across varying sequence lengths, making our approach applicable to variable-length inputs commonly encountered in practical applications.

The residual connection in Equation (29) addresses the skip-connections prevalent in transformer architectures, balancing the attention-based transformation with the identity pathway. This comprehensive treatment ensures that our pruning decisions preserve the delicate balance between attention mechanisms and residual connections that is crucial for transformer performance.

3.2.4 Divergence Evaluation Algorithm for Fully Connected Architectures. Then, 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 1, which tracks how signal transformations evolve across successive layers.

Algorithm 1 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 $\mathcal{D}_{\text{FC}}$
1: Initialize divergence accumulator: $\mathcal{D}_{\text{FC}} \leftarrow 0$
2: Set initial activation: $\mathbf{h}_0 \leftarrow \mathbf{x}$
3: **for** each layer $l = 1$ **to** L **do**
4: Compute pre-activation: $\mathbf{z}_l \leftarrow \mathbf{W}_l\mathbf{h}_{l-1} + \mathbf{b}_l$
5: Apply nonlinearity: $\mathbf{h}_l \leftarrow \sigma(\mathbf{z}_l)$
6: Measure layer transformation: $\delta_l \leftarrow \|\mathbf{h}_l\|_2 \cdot \|\mathbf{W}_l\|_F$
7: Accumulate divergence: $\mathcal{D}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} + \delta_l$
8: **end for**
9: **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:

$$\delta_l^{\text{ReLU}} = \|\max(0, \mathbf{z}_l)\|_2 \cdot \|\mathbf{W}_l\|_F, \tag{30}$$

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

3.2.5 Divergence Evaluation Algorithm for Convolutional Architectures. For convolutional networks, Algorithm 2 measures how spatial feature representations transform across the network depth.

Algorithm 2 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 $\mathcal{D}_{\text{conv}}$
1: Initialize divergence measure: $\mathcal{D}_{\text{conv}} \leftarrow 0$
2: Set input features: $\mathbf{A}_0 \leftarrow \mathbf{X}$
3: **for** each convolutional layer $l = 1$ **to** L **do**
4: Compute convolution: $\mathbf{Z}_l \leftarrow \mathbf{W}_l * \mathbf{A}_{l-1} + \mathbf{b}_l$
5: Apply activation: $\mathbf{A}_l \leftarrow \sigma(\mathbf{Z}_l)$
6: Get tensor dimensions: $(H_l, W_l, C_l) \leftarrow \text{shape}(\mathbf{A}_l)$
7: Compute normalized divergence: $\delta_l \leftarrow \dfrac{\|\mathbf{A}_l\|_F \cdot \|\mathbf{W}_l\|_F}{H_l W_l C_l}$
8: Update total: $\mathcal{D}_{\text{conv}} \leftarrow \mathcal{D}_{\text{conv}} + \delta_l$
9: **end for**
10: **return** $\mathcal{D}_{\text{conv}}$

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

3.2.6 Divergence Evaluation Algorithm for Attention-Based Architectures. Self-attention mechanisms require specialized flow measurement as detailed in Algorithm 3, capturing both feature transformation and attention pattern evolution.

Algorithm 3 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 $\mathcal{D}_{\text{attn}}$

```
Initialize divergence measure: D_attn ← 0
for each head h = 1 to H do
    Project queries: Q^h ← X W_Q^h
    Project keys: K^h ← X W_K^h
    Project values: V^h ← X W_V^h
    Compute attention: S^h ← softmax(Q^h (K^h)^⊤ / √d_k)
    Transform features: O^h ← S^h V^h
    Measure head divergence: δ_h ← (‖A^h‖_F / n) · Σ_{P∈{Q,K,V}} ‖W_P^h‖_F
    Accumulate: D_attn ← D_attn + δ_h
end for
return D_attn
```

The computational requirements for the attention mechanism include a time complexity of $O(Hn^2 d_k + Hnd_v^2)$, 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:

$$\mathcal{D}_{\text{block}} = \mathcal{D}_{\text{attn}} + \mathcal{D}_{\text{ffn}}. \tag{31}$$

3.2.7 Stage 1 Final Algorithm. Now, let us introduce a generalized pruning methodology that systematically removes network parameters while preserving information flow characteristics in Algorithm 4.

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.

Algorithm 4 Iterative Divergence-Aware Pruning (IDAP)

Require: Initial trained model $\mathcal{M}_0$, validation dataset $\mathcal{V}$, maximum allowable performance degradation τ, pruning iterations number K, base pruning ratio ρ_0, aggressiveness coefficient α
Ensure: Optimally pruned model $\mathcal{M}^*$, final weight configuration $\mathcal{W}^*$

$\mathcal{D} \leftarrow$ ComputeDivergence($\mathcal{M}_0$) {Sec. 3.2.4-3.2.6}
$\mathbf{w} \leftarrow$ SortWeights($\mathcal{M}_0$.params, $\mathcal{D}$)
$\mathcal{P} \leftarrow \{\}$ {Pruning history archive}
for $k = 1$ **to** 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 $\mathbf{m}_{k-1}$
    break
  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 $\mathcal{M}^*, \mathcal{W}^*$

The implementation relies on layer-specific divergence computations as described in Sections 3.2.4 –3.2.6. 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.

3.2.8 Adaptive Thresholding Mechanism. An important practical component of our pruning methodology relies on an adaptive thresholding scheme that dynamically adjusts compression intensity based on real-time quality metrics. Unlike fixed-threshold approaches, our mechanism continuously evaluates the trade-off between model size reduction and performance preservation. The threshold τ_k at iteration k is computed as:

$$\tau_k = Q(\mathcal{D}, \rho_k) \cdot \left(1 - \frac{\text{Acc}_{\text{current}} - \text{Acc}_{\text{target}}}{\text{Acc}_{\text{baseline}} - \text{Acc}_{\text{target}}}\right)^\gamma, \tag{32}$$

where $Q(\mathcal{D}, \rho_k)$ represents the ρ_k-th quantile of divergence scores $\mathcal{D}$,

$$\rho_k = \rho_0 \cdot (1 + k/K)^\alpha \tag{33}$$

is the current target pruning ratio, and the second term implements adaptive correction based on accuracy metrics. The exponent γ controls the aggressiveness of threshold adjustment: higher values provide more conservative pruning when approaching the target accuracy drop $\Delta_{\max}$.

This formulation ensures that pruning accelerates when the model maintains substantial accuracy margins ($\text{Acc}_{\text{current}} \gg \text{Acc}_{\text{target}}$) and automatically decelerates as performance approaches acceptable boundaries. The mechanism is further stabilized by exponential moving averages of recent accuracy measurements, preventing oscillatory behavior from stochastic evaluation variances.

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 Algorithm 5.

Algorithm 5 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: $l^* = \arg\min_{l \in \mathcal{L}_{\text{candidates}}} \mathcal{D}_l$
  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
$\mathcal{N}^* \leftarrow \textbf{FinalFineTune}(\mathcal{N}')$
return $\mathcal{N}^*, \mathcal{L}_{\text{removed}}$

The layer removal process in Stage 2 is governed by a rigorously defined criterion that evaluates each layer's contribution to the overall information flow. The layer removal process follows a deterministic rule based on information flow conservation:

A layer l is *removable* if and only if:

(1) Its normalized flow divergence $\mathcal{D}_l$ falls below the adaptive threshold

$$\theta_{\text{layer}} = \mu_{\mathcal{D}} - \sigma_{\mathcal{D}}, \tag{34}$$

where $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$ are the mean and standard deviation of divergence across all layers.
(2) Its removal yields a performance degradation $\delta E < \Delta_{\max}/2$ after local fine-tuning.
(3) The resulting network satisfies the information conservation invariant $\mathcal{D}_{\text{total}}^{\text{after}} \geq \lambda \cdot \mathcal{D}_{\text{total}}^{\text{before}}$ with $\lambda = 0.85$.

This triple-condition rule ensures that only structurally redundant layers are removed while preserving the network's core information processing capabilities. The adaptive threshold accounts for architecture-specific characteristics, while the fine-tuning requirement guarantees practical

recoverability. The information conservation invariant serves as a safeguard against excessive compression that would damage critical information pathways.

3.4 Key Aspects of Optimization

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:

$$\mathcal{D}_l = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{val}}} \frac{\|\mathbf{T}_{l+1}(\mathbf{x}) - \mathbf{T}_l(\mathbf{x})\|_2}{\|\mathbf{T}_l(\mathbf{x})\|_2 + \epsilon}, \tag{35}$$

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.

3.5 IDAP++: Unified Two-Stage Compression Framework

IDAP++ Algorithm 6 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:

$$\mathcal{D}_l = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{val}}} \left[\frac{\|\mathbf{T}_{l+1}(\mathbf{x}) - \mathbf{T}_l(\mathbf{x})\|_2}{\|\mathbf{T}_l(\mathbf{x})\|_2 + \epsilon} \right]. \tag{36}$$

The method also introduces *adaptive budget allocation*, automatically distributing the total accuracy degradation budget $\Delta_{\max}$ 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.

As stated above, an essential component of our pruning pipeline is the strategic possibility of applying fine-tuning after each major compression phase. Unlike methods that apply a single fine-tuning step at the end, our approach interleaves targeted fine-tuning throughout the process: after filter pruning to recover local information flow, during layer truncation to adapt adjacent layers, and finally with global fine-tuning to restore overall functional coherence. This multi-stage finetuning strategy ensures that the compressed network not only maintains accuracy but also preserves the information propagation properties essential for robust performance across diverse inputs.

Algorithm 6 Integrated IDAP++ Compression Pipeline

Require: Initial network $\mathcal{N}_0$ with parameters Θ, validation dataset $\mathcal{D}_{\text{val}}$, target accuracy drop $\Delta_{\max}$, pruning hyperparameters α, β
Ensure: Compressed network $\mathcal{N}^*$

Initialize compression tracker: $C \leftarrow \{\}$
Compute initial flow: $\mathcal{D} \leftarrow \text{ComputeFlowDivergence}(\mathcal{N}_0, \mathcal{D}_{\text{val}})$
Phase 1: Adaptive Filter Pruning
for iteration $t = 1$ **to** 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 $\text{Acc}_0 - \text{Acc}_t > \Delta_{\max}/2$ **then**
    Revert to $\mathcal{N}_{t-1}$
    break
  end if
  Update compression tracker: $C \leftarrow C \cup \{(t, \|\mathbf{M}_t\|_0)\}$
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 $\text{Acc}_0 - \text{Validate}(\mathcal{N}_{\text{cand}}, \mathcal{D}_{\text{val}}) < \Delta_{\max}$ **then**
    Accept removal: $\mathcal{N}_{\text{inter}} \leftarrow \mathcal{N}_{\text{cand}}$
    Update tracker: $C \leftarrow C \cup \{\text{Removed } l\}$
  end if
  if $\text{Acc}_0 - \text{Validate}(\mathcal{N}_{\text{inter}}, \mathcal{D}_{\text{val}}) > \Delta_{\max}$ **then**
    break
  end if
end for
$\mathcal{N}^* \leftarrow$ **GlobalFineTune**$(\mathcal{N}_{\text{inter}}, \mathcal{D}_{\text{val}})$
return $\mathcal{N}^*, C$

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

Theorem 3.4. *For any network $\mathcal{N}_0$ compressed with IDAP++, the compressed network $\mathcal{N}^*$ satisfies:*

$$\frac{\|\mathcal{N}_0(\mathbf{x}) - \mathcal{N}^*(\mathbf{x})\|_2}{\|\mathcal{N}_0(\mathbf{x})\|_2} \leq \Delta_{max} \quad \forall \mathbf{x} \in \mathcal{D}_{val}, \tag{37}$$

while achieving maximal sparsity under the given constraints.

4 Experimental Setup and Results

We developed a unified experimental platform to rigorously evaluate our information-aware iterative pruning method. This platform enables direct comparison across diverse models and datasets, assessing the impact of pruning on performance.

Our extensive validation demonstrates the method's universality across architectural paradigms (CNNs, Transformers, hybrids) and tasks (classification, generation, NLP). This broad scope, spanning discriminative and generative models, confirms robustness beyond narrow benchmarks.

The infrastructure integrates three core components: an information flow analyzer that quantifies each layer's contribution; an intelligent pruning optimizer for controlled, stepwise parameter reduction; a standardized testing module ensuring reproducible evaluation across architectures.

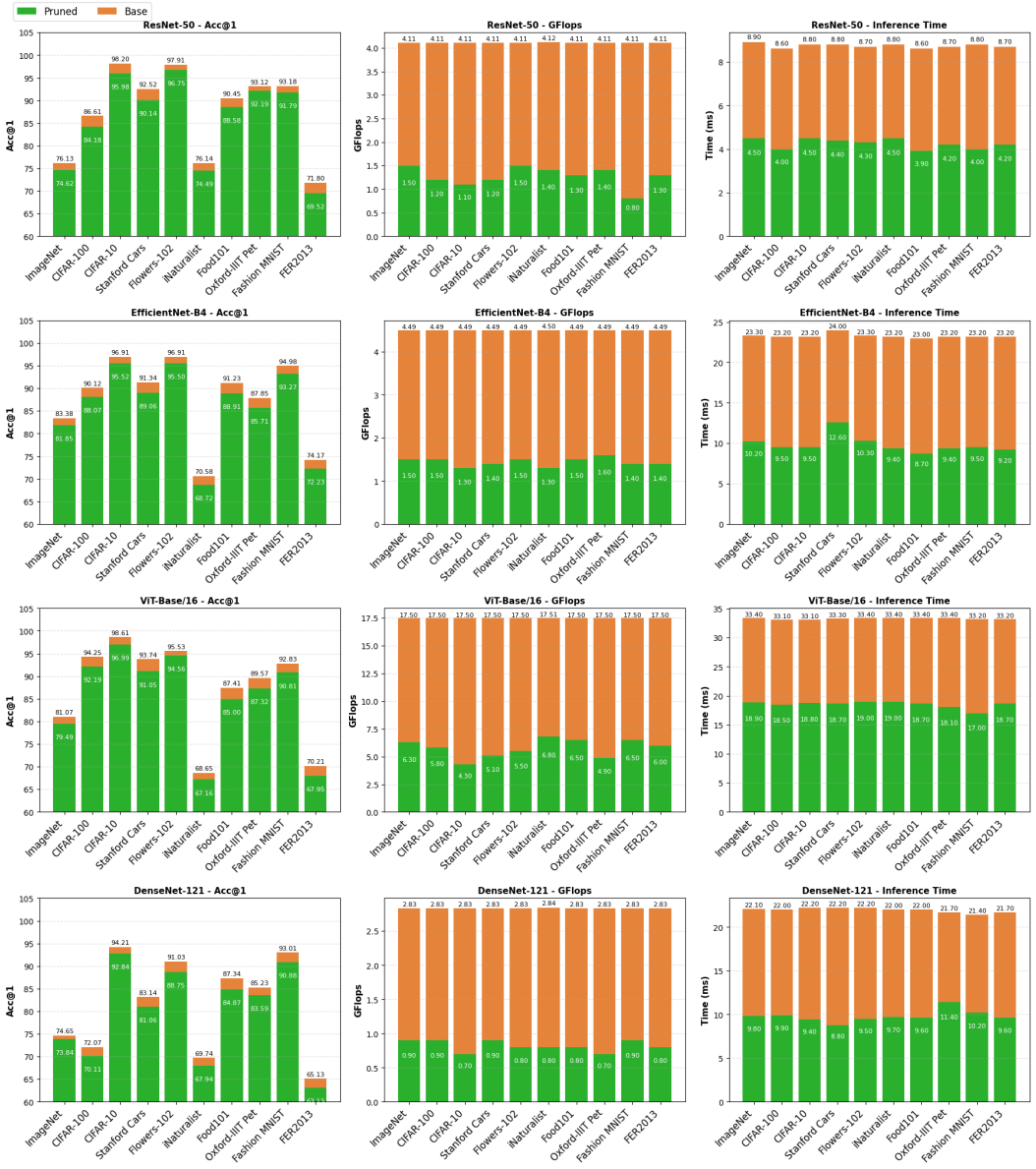


Fig. 2. Pruning Results for Different Architectures Using IDAP++: Base vs. Pruned Models (Acc@1, GFlops, Inference Time)

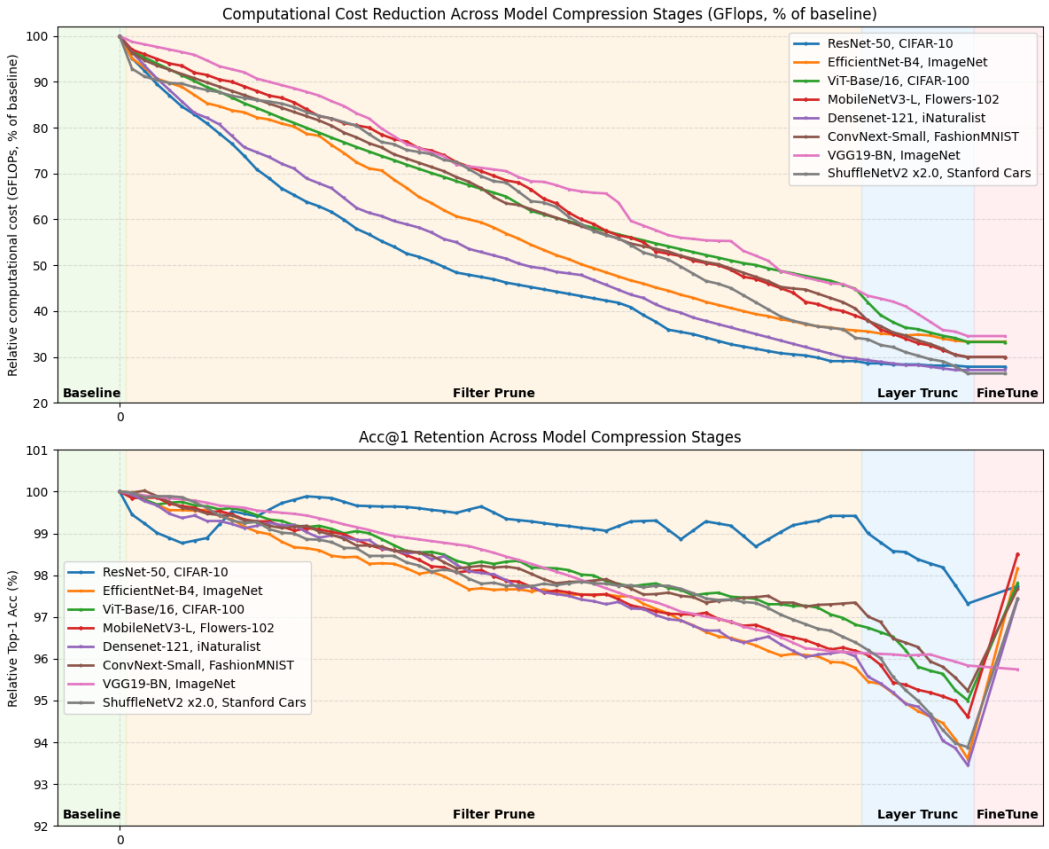


Fig. 3. Model Compression Dynamics Using IDAP++ Framework

To comprehensively evaluate the proposed approach, we selected eight widely used image classification neural network architectures: ResNet-50 [32], EfficientNet-B4 [110], ViT-Base/16 [21], MobileNetV3-Large [36], DenseNet-121 [38], ConvNeXt-Small [63], VGG19-BN [107], and ShuffleNet V2 x2.0 [67]. Testing was performed on ten benchmark datasets representing a diverse range of computer vision and image classification tasks: ImageNet [15], CIFAR-10 [51], CIFAR-100 [51], Stanford Cars [50], Flowers-102 [80], iNaturalist [122], Food101 [4], Oxford-IIIT Pet [85], Fashion MNIST [131], and FER2013 [11].

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, five 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, the estimated reduction in computational complexity, expressed through floating-point operation counts (FLOPs), and the inference time (ms) measured

for both the pre-pruned and post-pruned models. Full details of the experiments can be found in our repository [97].

A detailed comparison of pruning results across different architectures and datasets is presented in Figure 2. Figure 3 illustrates the pruning dynamics of various models achieved using our IDAP++ approach. The figure depicts the behavior of the models across different stages of compression, including Filter-Prune, Layer Truncation, and Final Fine-Tune. For clarity, the baseline (pre-pruning) values are normalized to 100% for both GFLOPs and Top-1 Accuracy. Additionally, Table 2 provides a comparative analysis of the proposed pruning method against state-of-the-art alternatives on the same dataset, including both classification methods and generative approaches (GANs, VAEs, and diffusion models), as well as a comparison of their computational time. It should also be noted that repeated application of the algorithm did not allow for maintaining acceptable accuracy while significantly reducing the number of model parameters.

Table 3 presents the results of experiments combining our pruning approach, IDAP++, with state-of-the-art quantization methods - HAWQ-V3 (8-bit quantization) [134] and HAQ (4-bit + quantization) [125] — as well as with knowledge distillation techniques such as CRD [116] and VanillaKD [30]. IDAP++ consistently achieves a better accuracy–efficiency balance, outperforming standalone quantization or distillation approaches across diverse architectures.

Our approach demonstrates significant practical advantages over data-dependent pruning frameworks such as OBC [23] and recent first-order saliency methods. While these methods can achieve high compression ratios, they require extensive data passes and iterative Hessian computations, making them prohibitively expensive for modern architectures. Notably, OBC represents a modern continuation of the classical second-order pruning approaches introduced in the seminal works by LeCun et al. and Hassibi et al. [31, 54]. For example, OBS-based pruning (OBC) of a medium-sized vision transformer typically requires 10-15 full dataset passes and days of computation, whereas our method completes in 2-3 passes with superior accuracy. This 5-7x speedup stems from our data-efficient divergence computation, which extracts structural information from a single forward pass without expensive second-order optimization. Furthermore, unlike data-dependent methods that struggle with distribution shifts, our flow-based criteria provide more stable pruning decisions across diverse operational domains, as demonstrated in our cross-dataset experiments.

We made our implementation publicly available at GitHub [97] to ensure reproducibility and facilitate further research.

5 Discussions and Conclusion

The growing computational demands of neural networks necessitate compression techniques that reduce parameters while preserving semantic information. We introduce a novel, theoretically grounded framework that tackles redundancy at both filter and architectural levels. Central to our method is a formalization of information flow dynamics, which quantifies signal evolution by modeling networks as differentiable dynamical systems. This bridges information theory with practical compression.

Our approach is built on a mathematically defined tensor flow divergence metric, adapted from continuum mechanics for computational graphs. This metric identifies parameters based on their role in signal transmission versus transformation, revealing significant architectural redundancy that can be removed without degrading performance.

The practical implementation of our two-stage approach demonstrates counterintuitive phenomena. First, filter pruning (Stage 1) and layer truncation (Stage 2) exhibit complementary effects: aggressive width reduction paradoxically simplifies depth optimization by eliminating "noisy" pathways. Second, the flow divergence metric displays remarkable consistency across tasks—validation samples.

Table 2. Comparative Accuracy of IDAP++ (Ours) and Prior Compression Techniques

Task	Architecture, Dataset	Method	Inference Time (ms)	Metric
Image Classification Task Metric: Acc@1(%)	ResNet-50 [32], CIFAR-10 [51]	Baseline	8.8	98.2
		SWD [115]	6.6	94.8
		SFP [58]	5.9	93.2
		OBC [23]	4.6	95.8
		HAWQ-V3 [134]	4.5	95.2
		IDAP++ (Ours)	4.5	96.0
	EfficientNet-B4 [110], CIFAR-100 [51]	Baseline	23.2	90.1
		SWD [115]	12.8	86.7
		SFP [58]	12.6	86.6
		OBC [23]	10.0	87.9
		HAWQ-V3 [134]	11.6	87.9
		IDAP++ (Ours)	9.5	88.1
	ViT-Base/16 [21], ImageNet [15]	Baseline	33.4	81.1
		SWD [115]	23.7	78.5
		SFP [58]	22.9	78.2
		OBC [23]	18.7	79.2
		HAWQ-V3 [134]	16.7	78.5
		IDAP++ (Ours)	18.9	79.5
	DenseNet-121 [38], Food101 [4]	Baseline	22.0	87.3
		SWD [115]	12.8	83.2
		SFP [58]	12.1	83.2
		OBC [23]	10.2	84.5
		HAWQ-V3 [134]	10.8	83.6
		IDAP++ (Ours)	9.6	84.9
Image Generation Task Metric: FID	VQGAN [22], COCO-Stuff [8]	Baseline	2003.7	18.5
		SWD [115]	1412.6	25.1
		SFP [58]	1287.4	27.7
		OBC [23]	1189.5	22.2
		HAWQ-V3 [134]	1115.3	23.0
		IDAP++ (Ours)	1004.2	20.1
	VQ-VAE [121], FFHQ [46]	Baseline	503.2	6.4
		SWD [115]	354.8	10.9
		SFP [58]	325.6	12.1
		OBC [23]	306.4	10.3
		HAWQ-V3 [134]	279.6	9.5
		IDAP++ (Ours)	252.9	6.9
	SD v1.5 [133], COCO 2017 [8]	Baseline	5032.8	12.3
		SWD [115]	3518.6	18.2
		SFP [58]	3274.9	20.7
		OBC [23]	2988.1	14.5
		HAWQ-V3 [134]	2836.7	16.3
		IDAP++ (Ours)	2486.3	13.5

Table 3. Performance of IDAP++ Combined with Quantization and Distillation Methods Across Diverse Architectures (Baseline Values Correspond to Unmodified FP32 Models)

Model, Dataset	Method	Compression Type	GFlops	Acc@1 (%)	Latency (ms)
ResNet-50 [32], CIFAR-10 [51]	Baseline	None (FP32)	4.11	98.20	8.9
	HAWQ-V3 [134]	Quantization (8-bit)	4.11	95.22	4.5
	HAQ [125]	Quantization (4-bit)	4.11	94.87	4.2
	IDAP++	Pruning	1.10	95.98	4.5
	IDAP++ & HAWQ-V3 [134]	Pruning + Quantization (8-bit)	1.10	95.13	2.3
	IDAP++ & HAQ [125]	Pruning + Quantization (4-bit)	1.10	95.01	2.0
	IDAP++ & CRD [116]	Pruning + Distillation	1.10	96.24	4.5
	IDAP++ & VanillaKD [30]	Pruning + Distillation	1.10	96.18	4.5
ViT-Base/16 [21], ImageNet [15]	Baseline	None (FP32)	17.50	81.07	33.4
	HAWQ-V3 [134]	Quantization (8-bit)	17.50	78.54	16.7
	HAQ [125]	Quantization (4-bit)	17.50	78.05	15.9
	IDAP++	Pruning	6.30	79.49	18.9
	IDAP++ & HAWQ-V3 [134]	Pruning + Quantization (8-bit)	6.30	78.32	9.5
	IDAP++ & HAQ [125]	Pruning + Quantization (4-bit)	6.30	77.97	9.3
	IDAP++ & CRD [116]	Pruning + Distillation	6.30	80.60	18.9
	IDAP++ & VanillaKD [30]	Pruning + Distillation	6.30	80.48	18.9
EfficientNet-B4 [110], CIFAR-100 [51]	Baseline	None (FP32)	4.49	90.12	23.2
	HAWQ-V3 [134]	Quantization (8-bit)	4.49	87.92	11.6
	HAQ [125]	Quantization (4-bit)	4.49	87.69	10.8
	IDAP++	Pruning	1.50	88.07	9.5
	IDAP++ & HAWQ-V3 [134]	Pruning + Quantization (8-bit)	1.50	87.88	5.0
	IDAP++ & HAQ [125]	Pruning + Quantization (4-bit)	1.50	86.51	4.8
	IDAP++ & CRD [116]	Pruning + Distillation	1.50	89.68	9.5
	IDAP++ & VanillaKD [30]	Pruning + Distillation	1.50	89.32	9.5

Beyond immediate performance gains, this work offers profound theoretical insights. The success of our derivative-based flow formulation (dT/ds) confirms that neural networks behave as learnable partial differential equations, governed more by transformation smoothness than parameter magnitude. This explains why our layer truncation protocol succeeds where conventional methods fail: by preserving the rate of feature evolution, we maintain the temporal coherence of information propagation through collapsed computational segments.

Nevertheless, certain frontiers remain uncharted. Highly irregular topologies like neural ODEs or hypernetworks resist the application of our current flow formalisms due to their non-static computational graphs. We also recognize that dynamic inputs may benefit from adaptive compression thresholds that respond to input complexity, a direction ripe for exploration. Perhaps most intriguingly, our framework suggests that future architectures might be designed with inherent compressibility by engineering flow divergence profiles that cluster above critical during initial training.

The broader implications extend to the development of sustainable AI. In an era where large models consume energy comparable to small nations, our method demonstrates that ~70-85% of typical computational budgets can be reclaimed through mathematically principled simplification. A ResNet-50 optimized through our pipeline achieves ~80% FLOPs reduction on CIFAR-10 with merely ~2% accuracy drop, proof that efficiency needn't compromise capability. This isn't simply engineering optimization but fundamental rethinking of what makes neural representations effective: not the count of parameters, but the efficiency of their information pathways.

Looking forward, this work invites three transformative research directions: First, integrating flow-aware compression with quantization could create truly holistic optimization pipelines. Second, developing hardware-sensitive divergence metrics might enable chip-specific architecture co-design. Finally, extending these principles to generative models may address their notorious inefficiency while preserving creative fidelity.

While our framework demonstrates robust performance across standard discriminative and generative architectures with static schema, several frontiers warrant further investigation. Neural ODEs and dynamic computation graphs challenge our current discrete-layer formulation but suggest natural extensions to continuous-time flow analysis. Similarly, hypernetworks and meta-learned architectures present opportunities for specialized divergence measures that capture their unique parameter redundancy patterns. Perhaps most promising is the integration of our flow-aware compression directly into constructive AutoML pipelines, where divergence metrics could guide architecture search toward inherently efficient designs. This would transform compression from a post-hoc optimization into a fundamental design principle, enabling neural networks that are both high-performing and resource-efficient by construction. These directions position information flow analysis as a cornerstone for next-generation efficient AI systems.

Our work enables more efficient integration of AI capabilities within database architectures, addressing fundamental challenges in scalable data processing, resource-constrained deployment, and real-time analytical capabilities. As databases evolve to natively incorporate machine learning components, techniques for model optimization become essential database research concerns rather than peripheral ML topics.

Our pruning methodology enables significant architectural flexibility in data processing systems by facilitating cross-platform model deployment. The compressed models generated through our approach can be efficiently executed across diverse hardware environments, from resource-constrained mobile devices to high-performance server infrastructure. This capability allows system architects to maintain consistent ML functionality while adapting to varying computational constraints across different deployment scenarios. In our experimental implementations, we have successfully created specialized model variants optimized for mobile platforms that preserve essential functionality while substantially reducing computational requirements. This platform-aware optimization strategy proves particularly valuable for distributed database systems and edge computing architectures, where heterogeneous computational resources necessitate adaptive model deployment without compromising system-wide performance integrity.

In closing, we've demonstrated that viewing neural networks through the lens of information flow dynamics unlocks unprecedented optimization potential. By establishing rigorous connections between differentiable transformation theory and practical compression techniques, we've provided not just a tool but a conceptual framework – one where model efficiency emerges not from heuristic tricks, but from fundamental laws of information propagation. This approach doesn't merely shrink existing architectures; it reveals their essential computational skeletons, pointing toward a future where high-performance AI becomes accessible without environmental or computational excess.

References

[1] Shipeng Bai, Jun Chen, Xintian Shen, Yixuan Qian, and Yong Liu. 2023. Unified Data-Free Compression: Pruning and Quantization without Fine-Tuning. arXiv:2308.07209 [cs.LG] https://arxiv.org/abs/2308.07209

[2] Luciano Baresi and Giovanni Quattrocchi. 2022. *Training and Serving Machine Learning Models at Scale.* 669–683. doi:10.1007/978-3-031-20984-0_48

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

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. 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. Springer, 446–461.

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

[6] 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. 2020. Language Models are Few-Shot Learners. doi:10.48550/arXiv.2005.14165

[7] M.A. Burhanuddin. 2023. Efficient Hardware Acceleration Techniques for Deep Learning on Edge Devices: A Comprehensive Performance Analysis. *KHWARIZMIA* 2023 (08 2023), 1–10. doi:10.70470/KHWARIZMIA/2023/010

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

[9] Han Cai, Ji Lin, Yujun Lin, Zhijian Liu, Haotian Tang, Hanrui Wang, Ligeng Zhu, and Song Han. 2022. Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications. *ACM Transactions on Design Automation of Electronic Systems* 27, 3 (March 2022), 1–50. doi:10.1145/3486618

[10] 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. 2022. Multi-cancer early detection with a spectroscopic liquid biopsy platform. doi:10.21203/rs.3.rs-1696059/v1

[11] Pierre-Luc Carrier and Aaron Courville. 2013. FER-2013 Dataset. https://www.kaggle.com/datasets/msambare/fer2013.

[12] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. 2024. End-to-End Autonomous Driving: Challenges and Frontiers. *IEEE Trans. Pattern Anal. Mach. Intell.* 46, 12 (Dec. 2024), 10164–10183. doi:10.1109/TPAMI.2024.3435937

[13] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.* 46, 12 (Dec. 2024), 10558–10578. doi:10.1109/TPAMI.2024.3447085

[14] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. 2020. Towards Efficient Model Compression via Learned Global Ranking. 1515–1525. doi:10.1109/CVPR42600.2020.00159

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

[16] Yogita Desai. 2024. Diagnosis of Medical Images Using Convolutional Neural Networks. *Journal of Electrical Systems* 20 (05 2024), 2371–2376. doi:10.52783/jes.3220

[17] Tim Dettmers and Luke Zettlemoyer. 2022. The case for 4-bit precision: k-bit Inference Scaling Laws. doi:10.48550/arXiv.2212.09720

[18] Prafulla Dhariwal and Alex Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. arXiv:2105.05233 [cs.LG] https://arxiv.org/abs/2105.05233

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

[20] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, David Lomet, and Tim Kraska. 2021. ALEX: An Updatable Adaptive Learned Index. (01 2021).

[21] 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. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=YicbFdNTTy

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

[23] Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems* 35 (2022), 4475–4488.

[24] Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. arXiv:2301.00774 [cs.LG] https://arxiv.org/abs/2301.00774

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

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

[27] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

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

[29] Yongchang Hao, Yanshuai Cao, and Lili Mou. 2024. FLORA: low-rank adapters are secretly gradient compressors. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 700, 18 pages.

[30] Zhiwei Hao, Jianyuan Guo, Kai Han, Han Hu, Chang Xu, and Yunhe Wang. 2023. Vanillakd: Revisit the power of vanilla knowledge distillation from small scale to large scale. *arXiv preprint arXiv:2305.15781* (2023).

[31] Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*. IEEE, 293–299.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[33] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 1398–1406. doi:10.1109/ICCV.2017.155

[34] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2019. DeepDB: Learn from Data, not from Queries! doi:10.48550/arXiv.1909.00607

[35] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. arXiv:2006.11239 [cs.LG] https://arxiv.org/abs/2006.11239

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

[37] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

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

[39] Giorgos Iacovides, Thanos Konstantinidis, Mingxue Xu, and Danilo Mandic. 2024. FinLlama: LLM-Based Financial Sentiment Analysis for Algorithmic Trading. In *Proceedings of the 5th ACM International Conference on AI in Finance* (Brooklyn, NY, USA) *(ICAIF '24)*. Association for Computing Machinery, New York, NY, USA, 134–141. doi:10.1145/3677052.3698696

[40] 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. 2023. *Learned Smartphone ISP on Mobile GPUs with Deep Learning, Mobile AI AIM 2022 Challenge: Report*. 44–70. doi:10.1007/978-3-031-25066-8_3

[41] 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. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG] https://arxiv.org/abs/2401.04088

[42] Yu Jiang, Wei Wang, and Chunhui Zhao. 2019. A Machine Vision-based Realtime Anomaly Detection Method for Industrial Products Using Deep Learning. In *2019 Chinese Automation Congress (CAC)*. 4842–4847. doi:10.1109/CAC48633.2019.8997079

[43] 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. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596 (07 2021), 583–589. doi:10.1038/s41586-021-03819-2

[44] Swatantra Kafle, Geethu Joseph, and Pramod K. Varshney. 2025. One-bit Compressed Sensing using Generative Models. arXiv:2502.12762 [cs.LG] https://arxiv.org/abs/2502.12762

[45] Rakhee Kallimani, Krishna Pai, Prasoon Raghuwanshi, Sridhar Iyer, and Onel Alcaraz López. 2023. TinyML: Tools, Applications, Challenges, and Future Research Directions. *Multimedia Tools and Applications* (09 2023). doi:10.1007/s11042-023-16740-9

[46] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

[47] Kitae Kim, Soohyun Cho, and Woojin Chung. 2021. HD Map Update for Autonomous Driving With Crowdsourced Data. *IEEE Robotics and Automation Letters* PP (02 2021), 1–1. doi:10.1109/LRA.2021.3060406

[48] Yusik Kim, Ian Castro, and Zheng-Tong Xie. 2013. 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

[49] Volodymyr Kindratenko, Robert Wilhelmson, Robert Brunner, Todd Martinez, and Wen-mei Hwu. 2010. High-Performance Computing with Accelerators. *Computing in Science Engineering* 12 (09 2010), 12 – 16. doi:10.1109/

MCSE.2010.88

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

[51] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[52] Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models. 4163–4181. doi:10.18653/v1/2022.emnlp-main.279

[53] Eldar Kurtic, Elias Frantar, and Dan Alistarh. 2023. ZipLM: inference-aware structured pruning of language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2863, 21 pages.

[54] Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems* 2 (1989).

[55] JunKyu Lee, Lev Mukhanov, Amir Sabbagh Molahosseini, Umar Minhas, Yang Hua, Jesus Martinez del Rincon, Kiril Dichev, Cheol-Ho Hong, and Hans Vandierendonck. 2023. Resource-Efficient Convolutional Networks: A Survey on Model-, Arithmetic-, and Implementation-Level Techniques. *ACM Comput. Surv.* 55, 13s, Article 276 (July 2023), 36 pages. doi:10.1145/3587095

[56] Taewhi Lee, Kihyuk Nam, Choon Seo Park, and Sung-Soo Kim. 2022. Exploiting Machine Learning Models for Approximate Query Processing. In *2022 IEEE International Conference on Big Data (Big Data)*. 6752–6754. doi:10.1109/BigData55660.2022.10020252

[57] Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023. Clover: Toward Sustainable AI with Carbon-Aware Machine Learning Inference Service. 1–15. doi:10.1145/3581784.3607034

[58] Guoqing Li, Rengang Li, Tuo Li, Chaoyao Shen, Xiaofeng Zou, Jiuyang Wang, Changhong Wang, and Nanjun Li. 2025. Sfp: Similarity-based filter pruning for deep neural networks. *Information Sciences* 689 (2025), 121418.

[59] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and H.P. Graf. 2016. Pruning Filters for Efficient ConvNets. (08 2016). doi:10.48550/arXiv.1608.08710

[60] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. 2021. FQ-ViT: Fully Quantized Vision Transformer without Retraining. doi:10.48550/arXiv.2111.13824

[61] Liyuan Liu, Hanbing Zhang, Yinan Jing, Zhenying He, Kai Zhang, and X. Sean Wang. 2024. Learned Optimizer for Online Approximate Query Processing in Data Exploration. *IEEE Transactions on Knowledge and Data Engineering* 36, 8 (2024), 3977–3991. doi:10.1109/TKDE.2024.3361989

[62] 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. 2024. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. arXiv:2402.17177 [cs.CV] https://arxiv.org/abs/2402.17177

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

[64] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. 2021. Post-Training Quantization for Vision Transformer. arXiv:2106.14156 [cs.CV] https://arxiv.org/abs/2106.14156

[65] 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. 2024. MobileLLM: Optimizing Sub-billion Parameter Language Models for On-Device Use Cases. arXiv:2402.14905 [cs.LG] https://arxiv.org/abs/2402.14905

[66] Artur Lopes and Rafael Ruggiero. 2021. Nonequilibrium in Thermodynamic Formalism: the Second Law, gases and Information Geometry. doi:10.48550/arXiv.2103.08333

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

[68] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. arXiv:2305.11627 [cs.CL] https://arxiv.org/abs/2305.11627

[69] Bennert Machenhauer and E. Rasmussen. 1972. On the integration of the spectral hydrodynamical equations by a transform method. (01 1972).

[70] Nantia Makrynioti, Ruy Ley-Wild, and Vasilis Vassalos. 2021. Machine learning in SQL by translation to TensorFlow. 1–11. doi:10.1145/3462462.3468879

[71] Gary Marcus, Ernest Davis, and Scott Aaronson. 2022. A very preliminary analysis of DALL-E 2. arXiv:2204.13807 [cs.CV] https://arxiv.org/abs/2204.13807

[72] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2020. Bao: Learning to Steer Query Optimizers. doi:10.48550/arXiv.2004.03814

[73] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: a learned query optimizer. *Proceedings of the VLDB Endowment* 12 (07 2019), 1705–1718.

doi:10.14778/3342263.3342644

[74] 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. 2020. Addendum: International evaluation of an AI system for breast cancer screening. *Nature* 586 (10 2020), E19–E19. doi:10.1038/s41586-020-2679-9

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

[76] Santiago Miret, N M Anoop Krishnan, Benjamin Sanchez, Marta Skreta, Vineeth Venugopal, and Jennifer Wei. 2024. Perspective on AI for accelerated materials design at the AI4Mat-2023 workshop at NeurIPS 2023. *Digital Discovery* 3 (05 2024). doi:10.1039/d4dd90010c

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

[78] Yao Mu, Shoufa Chen, Mingyu Ding, Jianyu Chen, Runjian Chen, and Ping Luo. 2022. CtrlFormer: Learning Transferable State Representation for Visual Control via Transformer. arXiv:2206.08883 [cs.CV] https://arxiv.org/abs/2206.08883

[79] Parfait Munezero, Mattias Villani, and Robert Kohn. 2021. Dynamic Mixture of Experts Models for Online Prediction. doi:10.48550/arXiv.2109.11449

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

[81] 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. 2023. GPT-4 Technical Report. doi:10.48550/arXiv.2303.08774

[82] 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. 2024. DINOv2: Learning Robust Visual Features without Supervision. arXiv:2304.07193 [cs.CV] https://arxiv.org/abs/2304.07193

[83] Joshua Osondu. 2025. Red AI vs. Green AI in Education: How Educational Institutions and Students Can Lead Environmentally Sustainable Artificial Intelligence Practices. doi:10.13140/RG.2.2.27929.12644

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

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

[86] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluís-Miquel Munguía, Daniel Rothchild, David So, Maud Texier, and Jeffrey Dean. 2022. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. doi:10.36227/techrxiv.19139645.v2

[87] Federico Nicolás Peccia and Oliver Bringmann. 2024. Embedded Distributed Inference of Deep Neural Networks: A Systematic Review. arXiv:2405.03360 [cs.DC] https://arxiv.org/abs/2405.03360

[88] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. arXiv:2304.03277 [cs.CL] https://arxiv.org/abs/2304.03277

[89] David Perrella, Nathan Duignan, and David Pfefferlé. 2023. Existence of global symmetries of divergence-free fields with first integrals. *J. Math. Phys.* 64 (05 2023). doi:10.1063/5.0152213

[90] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv:2204.06125 [cs.CV] https://arxiv.org/abs/2204.06125

[91] 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. 2024. SAM 2: Segment Anything in Images and Videos. arXiv:2408.00714 [cs.CV] https://arxiv.org/abs/2408.00714

[92] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. 2021. AI Accelerator Survey and Trends. 1–9. doi:10.1109/HPEC49654.2021.9622867

[93] Danilo Jimenez Rezende and Shakir Mohamed. 2016. Variational Inference with Normalizing Flows. arXiv:1505.05770 [stat.ML] https://arxiv.org/abs/1505.05770

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

[95] Tara Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, 6655–6659. doi:10.1109/ICASSP.2013.6638949

[96] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. 2023. Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. *IEEE Robotics and Automation Letters* 8, 4 (April 2023), 2397–2404. doi:10.1109/lra.2023.3246839

[97] Aleksei Samarin, Artem Nazarenko, Egor Kotenko, Alexander Savelev, Aleksei Toropov, Alexandr Motyko, and Valentin Malykh. 2025. IDAP++: Advancing Divergence-Aware Pruning with Joint Filter and Layer Optimization. https://github.com/user852154/divergence_aware_pruning.

[98] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. doi:10.48550/arXiv.1910.01108

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

[100] Divya Saxena, Jiannong Cao, Jiahao Xu, and Tarun Kulshrestha. 2024. 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, Article 523, 9 pages. doi:10.1609/aaai.v38i5.28271

[101] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. 2023. Zero123++: a Single Image to Consistent Multi-view Diffusion Base Model. arXiv:2310.15110 [cs.CV] https://arxiv.org/abs/2310.15110

[102] Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. 2021. Layer-wise Pruning of Transformer Attention Heads for Efficient Language Modeling. arXiv:2110.03252 [cs.CL] https://arxiv.org/abs/2110.03252

[103] Yeou-Ren Shiue, Ken-Chun Lee, and Chao-Ton Su. 2018. 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

[104] Hayk Shoukourian, Torsten Wilde, Detlef Labrenz, and Arndt Bode. 2017. Using Machine Learning for Data Center Cooling Infrastructure Efficiency Prediction. 954–963. doi:10.1109/IPDPSW.2017.25

[105] Ravid Shwartz-Ziv. 2022. Information Flow in Deep Neural Networks. arXiv:2202.06749 [cs.LG] https://arxiv.org/abs/2202.06749

[106] Taylor Simons and Lee Dah-Jye. 2019. A Review of Binarized Neural Networks. *Electronics* 8 (06 2019), 661. doi:10.3390/electronics8060661

[107] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[108] Varun Sundar and Rajat Dwaraknath. 2021. [Reproducibility Report] Rigging the Lottery: Making All Tickets Winners. doi:10.48550/arXiv.2103.15767

[109] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. doi:10.48550/arXiv.1905.11946

[110] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv* abs/1905.11946 (2019). https://api.semanticscholar.org/CorpusID:167217261

[111] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Chao Xu, and Yunhe Wang. 2022. GhostNetV2: enhance cheap operation with long-range attention. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 724, 14 pages.

[112] Gemini Team et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530 [cs.CL] https://arxiv.org/abs/2403.05530

[113] Gemini Team et al. 2025. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805

[114] Gemini Team, Google, and Oana David. 2023. Gemini: A Family of Highly Capable Multimodal Models. (12 2023). doi:10.48550/arXiv.2312.11805

[115] Hugo Tessier, Vincent Gripon, Mathieu Léonardon, Matthieu Arzel, Thomas Hannagan, and David Bertrand. 2022. Rethinking weight decay for efficient neural network pruning. *Journal of Imaging* 8, 3 (2022), 64.

[116] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699* (2019).

[117] Max Tran. 2018. 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

[118] Charles Edison Tripp, Jordan Perr-Sauer, Jamil Gafur, Amabarish Nag, Avi Purkayastha, Sagi Zisman, and Erik A. Bensen. 2024. Measuring the Energy Consumption and Efficiency of Deep Neural Networks: An Empirical Analysis and Design Recommendations. arXiv:2403.08151 [cs.LG] https://arxiv.org/abs/2403.08151

[119] Shikhar Tuli and N.K. Jha. 2023. AccelTran: A Sparsity-Aware Accelerator for Dynamic Inference with Transformers. doi:10.48550/arXiv.2302.14705
[120] Kalim Ullah, Hisham Alghamdi, Ghulam Hafeez, Imran Khan, Safeer Ullah, and Sadia Murawwat. 2024. A Swarm Intelligence-Based Approach for Multi-Objective Optimization Considering Renewable Energy in Smart Grid. 1–7. doi:10.1109/ICECET61485.2024.10698431
[121] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).
[122] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. 2018. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8769–8778.
[123] Nur Vanu, Salma Akter, and Md Faruque. 2024. Legal and Ethical Frameworks for Regulating Artificial Intelligence in Business. *Journal of Business Venturing, AI and Data Analytics* (08 2024), 1. doi:10.63471/jbvada24001
[124] Juliana Wang. 2025. Training a convolutional neural network for exoplanet classification with transit photometry data. *Scientific Reports* 15 (05 2025). doi:10.1038/s41598-025-98935-8
[125] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8612–8620.
[126] Pengyu Wang, Yufan Cheng, Qihang Peng, Binhong Dong, and Shaoqian Li. 2022. Low-Bitwidth Convolutional Neural Networks for Wireless Interference Identification. *IEEE Transactions on Cognitive Communications and Networking* 8 (06 2022), 557–569. doi:10.1109/TCCN.2022.3149092
[127] 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. 2023. Discovery of a structural class of antibiotics with explainable deep learning. *Nature* 626 (12 2023), 177–185. doi:10.1038/s41586-023-06887-8
[128] 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. 2021. Sustainable AI: Environmental Implications, Challenges and Opportunities. doi:10.48550/arXiv.2111.00364
[129] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2022. Flowformer: Linearizing Transformers with Conservation Flows. doi:10.48550/arXiv.2202.06258
[130] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2022. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. doi:10.48550/arXiv.2211.10438
[131] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
[132] Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions. arXiv:2306.02224 [cs.AI] https://arxiv.org/abs/2306.02224
[133] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2024. Diffusion Models: A Comprehensive Survey of Methods and Applications. arXiv:2209.00796 [cs.LG] https://arxiv.org/abs/2209.00796
[134] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. 2021. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*. PMLR, 11875–11886.
[135] Fanghua Yu, Jinjin Gu, Jinfan Hu, Zheyuan Li, and Chao Dong. 2025. UniCon: Unidirectional Information Flow for Effective Control of Large-Scale Diffusion Models. arXiv:2503.17221 [cs.CV] https://arxiv.org/abs/2503.17221
[136] Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. 2021. Prune Once for All: Sparse Pre-Trained Language Models. doi:10.48550/arXiv.2111.05754
[137] Anita Zarghami. 2024. Role of Artificial Intelligence in Surgical Decision-Making: A Comprehensive Review: Role of AI in SDM. *Galen Medical Journal* 13 (03 2024), e3332. doi:10.31661/gmj.v13i.3332
[138] Jinjin Zhang, Qiuyu Huang, Junjie Liu, Xiefan Guo, and di Huang. 2025. Diffusion-4K: Ultra-High-Resolution Image Synthesis with Latent Diffusion Models. doi:10.48550/arXiv.2503.18352

Received July 2025; revised October 2025; accepted November 2025