AI

MDPI

Review

Dynamic Inference Techniques in Convolutional Neural Networks: A Review

Joe Kareh [1] and Rodrigue Imad [1,2,*]

[1] Department of Computer Engineering, University of Balamand, Tripoli P.O. Box 100, Lebanon; joe.kareh@std.balamand.edu.lb
[2] Lab-STICC, UMR CNRS 6285, 29238 Brest, France
* Correspondence: rodrigue.imad@balamand.edu.lb

Abstract

The demand for more accurate Convolutional Neural Networks (CNNs) has led to increasingly deep and computationally expensive models. These architectures face limiting deployment and resource constraints when implemented. Static CNN models treat all input samples with identical computational density. These models are inefficient given the variability of real-world data. Dynamic inference solutions focus on this challenge by presenting computational graphs on a per-input basis. This allows considerable optimization of the efficiency–accuracy trade-off. This review offers a structured synthesis of dynamic inference solutions in CNNs, distributed over three architectural dimensions: depth, width, and spatial resolution. For each aspect, we present representative models, alongside their training strategies, and examine their practical constraints and performance trade-offs. We further present multi-dimensional hybrid architectures based on combining these aspects for accumulated efficiency gains. By critically analyzing each approach and their challenges—including hardware limitations, training instability, and constrained deployment scope—this paper offers a roadmap for future work in optimized deep learning modelling.

Keywords: conditional computation; convolutional neural networks (CNNs); computational efficiency; dynamic inference; dynamic convolution; dynamic width; early exiting; layer skipping; multi-dimensional inference; spatial resolution

1. Introduction

Convolutional Neural Networks (CNNs) are at the core of computer vision tasks, allowing several breakthroughs in image classification, object detection, and semantic segmentation [1–3]. These models' efficiency comes from a hierarchical architecture inspired by the human visual cortex, which allows independent feature extraction based on raw input pixel data across several abstraction levels [4]. Typical CNN models present multiple convolutional layers, which extract spatial features using trainable filters; pooling layers, which shrink spatial dimensionality; and fully connected layers, responsible for producing the final classification result [4,5]. This architecture, evolved through revolutionary CNN models such as LeNet [4], AlexNet [2], and VGGNet [6], was able to reach high accuracy on benchmark datasets. Nevertheless, this accuracy comes at high computational expenses, with significant memory bandwidth constraints and long inference latency [7]. These costs limit the deployment of state-of-the-art models in resource-constrained environments such as mobile computing, IoT applications, and autonomous systems [7,8].

Check for updates

Academic Editor: Kunhong Liu

Received: 18 August 2026
Revised: 28 September 2026
Accepted: 30 September 2026
Published: 3 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

This has inspired in-depth research into more optimized deep neural network architectures [9,10]. Classical frameworks rely on static compression—pruning, quantization, and low-rank decomposition [11–13]—providing constant, smaller networks, applied uniformly to all input data [14–16]. These solutions share a common constraint, detailed later on in Section 2.1: they all proceed with the same computation regardless of the treated sample complexity, generating unneeded redundancy for simple ones [7,17–19].

To resolve this inefficiency, a fundamental framework has evolved in what is called dynamic inference [20]. In contrast to static convolutional models, dynamic ones can adapt their computational graphs derived at inference time from each input data properties [7,21]. The objective of these models is to match computational cost with input complexity, reducing unnecessary calculations while preserving or enhancing accuracy [20]. By adjusting computation per input difficulty, dynamic models achieved better efficiency–accuracy trade-offs over a wide range of real-world applications [22].

The field of dynamic CNN inference encompasses a diverse range of techniques targeting different forms of computational redundancy [20]. These strategies are broadly classified based on the dynamic architectural dimensions they implement:

- Dynamic Depth: Models scale the number of executed layers on a per-input basis. Representative architectures include BranchyNet [23], SkipNet [24], BlockDrop [25], and ConvNet-AIG [21].
- Dynamic Width: Methods adjust the number of active channels and filters to match input complexity, as demonstrated by LC-Net [7] and dynamic convolution [26].
- Dynamic Spatial Resolution: These approaches focus on the spatial overlap in activation maps and input data [27,28], as shown in RANet [17], AR-Net [29], and spatially adaptive inference architectures [30,31].
- Multi-Dimensional Hybrid Strategies: Frameworks such as the Dual Dynamic Inference (DDI) [8] merge multiple dynamic dimensions for aggregated efficiency gains.

This paper is, by choice, narrower in scope than most general surveys of dynamic networks, such as the work of Han et al. [20]. This paper surveys and classifies the field across three axes, instance, space and time, spanning CNNs, sequential models, and NLP. We restrict our scope mainly to vision inference in CNNs and classify it across three different axes than Han et al. [20]: depth, width, and spatial resolution. This structure is intended to map directly onto design choice rather than the adaptivity mechanism in the abstract. This paper's scope offers a comparative treatment and a special focus on the gap between theoretical hardware gains in terms of hardware efficiency (Sections 4.3 and 7), which this paper treats as a throughline rather than an isolated objective.

This paper offers a comprehensive and critical review of dynamic inference in CNNs, providing a structured categorization and analysis of methods spanning from early exiting to spatial resolution. The remainder of this paper is organized as follows. Section 2 establishes the conceptual background of dynamic inference. Sections 3–6 discuss dynamic depth, width, spatial resolution, and multi-dimensional hybrid approaches, respectively, each with a focus on representative architectures, their limitations, and comparative analysis. Section 7 identifies open challenges and future directions. Section 8 concludes with a summary of key lessons learned.

This work's literature was gathered by combining targeted keyword searches on standard academic databases such as Google Scholar, IEEE Xplore, and arXiv covering recent works, in addition to citation-chaining from key seminal works as well as the authors' expertise in collecting relevant research for each dynamic-inference category and its representative architectures. Our work does not present any systematic review based on any protocol. It was purely guided by relevance to the dynamic inference's three main dimensions, as well as the work's influence.

Table 1 summarizes the main strengths and limitations of representative CNN architectures in the context of computational efficiency. While these architectures established important advances in accuracy and network design, their fixed computational structure applies essentially the same processing to every input. These limitations motivate the development of dynamic inference approaches.

Table 1. Representative CNN architectures: strengths and weaknesses motivating dynamic inference.

CNN	Strength	Weakness (Motivating Dynamic Inference)
LeNet [4]	Simple, efficient, proved CNN viability	Too shallow for complex modern tasks
AlexNet [2]	First deep CNN at ImageNet scale; established convolutional hierarchy	High parameter count; uniform computation regardless of input
VGGNet [6]	Simple, uniform 3 × 3 conv design; strong accuracy	Very high FLOPs/parameters for its accuracy; heavily over-parameterized
ResNet [3]	Residual connections enable very deep training; strong accuracy-depth trade-off	Fixed depth executed identically for every input, regardless of difficulty
MobileNet (V1–V3) [32–34]	Depthwise-separable convs; designed for mobile/edge efficiency	Still static—full network executed uniformly; depthwise convs are FLOPs-light but often memory-bound on GPU

2. Background and Preliminaries

This section establishes the conceptual foundations of dynamic inference, providing the background necessary for the analysis in Sections 3–6. We begin by comparing static and dynamic inference, and then introduce the principle of conditional computation. We then examine input difficulty variability, the primary motivation behind dynamic inference. Then, we will focus on the objective of dynamic inference which is the optimization of the accuracy–efficiency trade-off in these discussed networks and models. Finally, we introduce the three architectural dimensions along which dynamic inference operates, forming the organizing framework for this review.

2.1. Static Versus Dynamic Inference

CNNs have achieved strong performance across vision tasks including image classification and object detection. This success has been mainly built on the single pattern of static inference on the input data level [4,35]. The computational graph of these networks is fixed at training time and remains unchanged during deployment, regardless of the input being processed [7]. Whether the model is processing a high-quality, high-resolution, clear image of a single object, or a noisy scene with a cluttered desired object, the model employing static inference utilizes its full capacity equally, computing each convolutional filter it holds, including all activation functions and dense layers, applied identically to every input. This is what we call the "one-size-fits-all" strategy. In contrast to it, dynamic inference adjusts computational load per input at runtime, choosing only layers and operations needed on a case-by-case basis while preserving elevated accuracy [21].

2.2. Accuracy–Efficiency Trade-Off

The main objective of any dynamic inference architecture is not just to bound computational effort indiscriminately, but to optimize and reorganize the basic trade-off between the efficiency of this architecture and its output accuracy. The goal then is to construct a specific network featuring the property of operating at several points based on this trade-off curve, wisely choosing the optimal concurring point for each input sample. This efficient

ability presents a flexible capability easily able to outstand compression approaches in static network architectures such as quantization and pruning, which construct a fixed and inefficient model concentrating its focus over only a single point of the curve [7].

The benefits of dynamic inference are measured across several interconnected indicators:

- Computational Cost: Measured in FLOPs or MAC operations, this indicator captures the raw computational load of the model per input.
- Inference Latency: Wall-clock time per input sample is the most practically relevant metric for real-world deployment. It is shaped by hardware architecture, memory management, and software optimization [7].
- Task Accuracy: Dynamic models aim to match or exceed the predictive accuracy of their static counterparts, measured via standard metrics such as Top-1/Top-5 accuracy or mean Average Precision (mAP) [7,36].
- Energy Consumption: Energy expenditure is a critical constraint for battery-powered devices. By eliminating unnecessary operations, dynamic architectures reduce GPU power consumption, making them well-suited for edge computing and IoT applications [7,36].

Thus, we can formulate the objective of dynamic inference in convolutional neural networks as follows. Given an input sample $x \in \mathbb{R}^{H\times W\times C}$, let $p(x)$ denote a policy that selects a computational path $C_{p(x)}$ from the set of all possible execution paths in the network. The function $f(x; C_{p(x)})$ represents the task performance achieved when processing x through the selected path, and $\text{Cost}(C_{p(x)})$ quantifies the associated computational expenditure. Cost is not related to a unique metric: based on the deployment objective, it can be defined as the number of FLOPs/MAC, measure inference delay, or energy consumption.

The number of FLOPs is used by default as the comparison metric from Sections 3–6 to compare the cited works, while the distinction between theoretical and measured latency is addressed in Sections 4.3 and 7. The optimal policy $p^*(x)$ minimizes this cost while ensuring that performance does not degrade beyond a tolerance ε relative to the static baseline $f_{\text{static}}(x)$. Formally,

$$p^*(x) = \arg\min_p \text{Cost}(C_{p(x)}) \text{ s.t. } f(x; C_{p(x)}) \geq f_{\text{static}}(x) - \varepsilon \quad (1)$$

Equation (1) is intended as a general conceptual formulation rather than a complete multi-objective optimization model. The term $\text{Cost}(\cdot)$ may represent different deployment-dependent efficiency measures, including FLOPs, latency, or energy consumption. In practical dynamic inference systems, additional deployment factors and metrics—such as routing and gating overhead, memory-access patterns, hardware utilization, and throughput—may also affect realized efficiency. These aspects are not represented explicitly in Equation (1), but should be considered when evaluating dynamic models on real-world deployment platforms.

The architectural dimensions of dynamic inference—depth, width, and spatial resolution—are not mutually exclusive. The Dual Dynamic Inference (DDI) framework [8] demonstrates this by combining layer-wise (depth) and channel-wise (width) skipping in parallel, showing that the largest efficiency gains arise from applying dynamism across multiple axes simultaneously.

2.3. Design Aspects of Dynamic Inference

To realize conditional computation, adaptability must be built into one or more of the network's primary architectural dimensions: depth, width, and resolution. This categorization provides the organizing framework for Sections 3–6.

1. Dynamic Depth: Dynamic depth adapts the number of executed layers per input—the most intuitive form of conditional computation. Easy inputs can be classified reliably from features extracted at early layers, making full network traversal unnecessary.
    a. Early Exiting, as in BranchyNet [23], attaches auxiliary classifiers to intermediate layers. A confidence threshold determines whether an input exits at each branch, bypassing deeper layers when confidence is sufficient. This approach is well-suited for anytime prediction scenarios.
    b. Layer Skipping, as in SkipNet [24] and BlockDrop [25], learns a gating function that determines whether to execute or skip each block. ConvNet-AIG [21] extends this to a fully adaptive inference graph, enabling non-sequential routing to the most relevant layers per input.
2. Dynamic Width: Rather than skipping entire layers, dynamic width methods selectively activate only a subset of filters within each layer, targeting the redundancy present within feature maps.
    a. Filter Skipping: In LC-Net [7], an auxiliary control network (C-Net) predicts which filters are needed per input, generating a binary activation mask for each layer.
    b. Dynamic Convolutions: Chen et al. [26] replace each standard convolutional layer with N parallel input-dependent filters, aggregated via learned attention weights, increasing representational capacity without modifying the network's structural dimensions.
3. Dynamic spatial resolution: This dimension targets spatial redundancy in images and feature maps, based on the observation that not all spatial regions contribute equally to the final prediction.
    a. Input Resolution Scaling: RANet [17] and AR-Net [29] dynamically select the resolution for each input, directing simple inputs through low-resolution paths while routing complex inputs through higher-resolution ones.
    b. Spatial Feature Sparsity: Methods such as [30,31] restrict computation to informative spatial locations within convolutional feature maps, bypassing convolutions entirely at uninformative positions.

In summary, a single objective motivates dynamic inference: minimizing computational redundancy by aligning resource allocation with input difficulty. This objective is implemented across three main architectural dimensions, depth, width, and spatial resolution, while hybrid approaches combine multiple dimensions. Table 2 summarizes these dynamic inference dimensions, their main mechanisms, representative works, and key objectives.

Table 2. Summary of dynamic inference dimensions, mechanisms, and representative works.

Dynamic Dimension	Sub-Strategy	Core Mechanism	Representative Work(s)	Key Objective
Dynamic Depth	Early Exiting	Auxiliary classifiers at intermediate layers with confidence thresholding	BranchyNet [23], MSDNet [22]	Exit easy inputs early, saving all remaining layers
Dynamic Depth	Layer Skipping	Learned gating function decides per-block execution	SkipNet [24], BlockDrop [25], ConvNet-AIG [21]	Non-sequential routing for fine-grained depth control

Table 2. *Cont.*

Dynamic Dimension	Sub-Strategy	Core Mechanism	Representative Work(s)	Key Objective
Dynamic Width	Filter/Channel Gating	Auxiliary network generates a per-input binary mask for filters	LC-Net [7], Runtime Pruning [37]	Prune redundant channels per input, reduce FLOPs
Dynamic Width	Dynamic Convolution	Attention-weighted aggregation of N parallel filters	Dynamic Conv [26]	Increase representational capacity without depth/width cost
Dynamic spatial resolution	Input Scaling	Policy network selects optimal resolution level per input	RANet [17], AR-Net [29]	Route easy inputs to lightweight low-resolution sub-networks
Dynamic spatial resolution	Spatial Sparsity	Gating or stochastic sampling applied to feature map regions	Verelst et al. [30], Xie et al. [31], SACT [18]	Skip convolution at uninformative spatial locations
Multi-Dimensional	Hybrid Dynamism	Joint optimization across depth, width, and/or resolution axes	DDI [8], OFA [38], DS-Net [39]	Compounded efficiency gains exceeding any single-axis method

2.4. Training Challenges and Respective Solutions

Implementing dynamic inference introduces a fundamental training challenge: routing decisions—whether to skip a layer or prune a channel—are discrete and non-differentiable. These binary choices cannot be directly optimized through standard gradient-based backpropagation. Several approaches have been developed to address this:

1. Reinforcement Learning: Framing routing decisions as a Markov Decision Process (MDP) enables reinforcement learning to train the policy network. BlockDrop [25] uses a reward function that encourages accuracy while penalizing the number of executed blocks.
2. Gumbel–Softmax Relaxation: Gumbel–Softmax offers a continuous approximation of discrete sampling [40], enabling gradient flow during backpropagation. It has been employed as a solution in Dynamic Convolutions [30] and Spatially Adaptive Inference [31].
3. Hybrid Learning: Some architectures combine supervised learning for classification with reinforcement learning for routing [24,25], exploiting the advantages of both frameworks.
4. Multi-Exit Network Training: Deploying multi-exit models requires specialized training to prevent gradient conflicts among classifiers at multiple exit points. Li et al. [41] address this with a strategy that balances gradient contribution across exits, integrated with knowledge distillation to enhance inter-classifier cooperation [42–44].

These architectures present different aspects of stability and scalability when compared: reinforcement learning exhibits unbiased minimization of the true objective, while also presenting high-variance gradient approximations and sample inefficiency. This requires constant reward shaping in order for the result to converge reliably. Applying Gumbel–Softmax enables lower variance to be attained with fully differentiable gradients, while training models faster and more stably. Hybrid strategies try to merge reinforcement learning's policy quality with a supervised warm start training stability. This, however,

comes at the cost of a more complicated, multi-stage training pipeline. Practice shows a wider adoption for Gumbel–Softmax-based strategies in more recent multi-dimensional architectures (Section 6), being more adequate for scaling training dynamics.

3. Dynamic Depth Inference

Dynamic depth inference scales the number of executed layers per input. It is the most intuitive form of conditional computation: a large fraction of easy inputs can be correctly classified from features extracted at early layers, making full network traversal redundant [22,23]. By constructing shorter computational paths for such inputs, dynamic depth methods reduce both FLOPs and inference latency without requiring any compromise in accuracy.

3.1. Early Exiting

Early exiting networks—also called multi-exit or branchy networks—are straightforward yet effective solutions for dynamic depth inference [23,45]. The technique attaches auxiliary classifiers—exit branches—to intermediate network layers. At each exit point, the auxiliary classifier evaluates prediction confidence. If confidence exceeds a preset threshold, inference terminates at that branch, returning the current prediction and bypassing all deeper layers.

During the inference process, each input sample is processed sequentially. At each intermediate exit point, an auxiliary classifier evaluates the confidence level of the current prediction. If the computed confidence exceeds a predefined threshold, inference is terminated and the prediction recorded at that exit is returned as the final output, bypassing all subsequent deeper layers. This makes early exiting well-suited for anytime prediction, where results must be produced within a strict time budget [46].

BranchyNet [23] is the foundational work in early exit inference, attaching exit branches to AlexNet and LeNet architectures. Its key contribution is a confidence thresholding mechanism based on the maximum softmax probability at each exit. On MNIST, their AlexNet variant achieves over 75% of test inputs exiting at the second branch, reducing computation by more than half while maintaining full-network accuracy [23]. Figure 1 illustrates the early-exit architecture of BranchyNet.

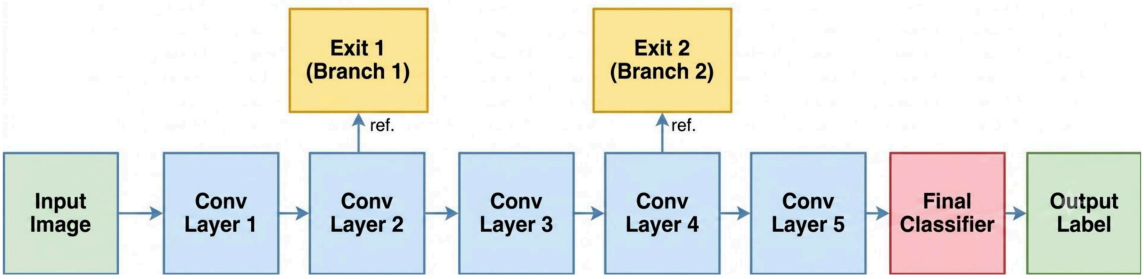


Figure 1. BranchyNet early-exit architecture. Auxiliary classifiers at intermediate layers allow easy inputs to exit early based on confidence thresholds, bypassing deeper layers and saving significant computation [23].

Subsequent work focused on improving the training of multi-exit architectures. A central challenge is gradient conflict: classifiers at different depths impose competing learning signals on the shared feature extractor. Li et al. [41] address this with a gradient equilibrium algorithm that dynamically balances gradient contributions from all exits during backpropagation, preventing any single exit from dominating the shared feature extractor. Their One-for-All Knowledge Distillation uses the deepest classifier's predictions as soft targets to guide earlier, weaker exits [41], improving consistency across exits and

strengthening the overall accuracy–efficiency trade-off on ImageNet and CIFAR. Building on this direction, JEI-DNN jointly learns the exit policy and the underlying classifier within a unified probabilistic framework, improving the calibration of exit decisions and reducing the mismatch between predicted and actual confidence at each exit [47].

Early exiting has been extended beyond image classification. MSDNet [22] is designed specifically for inference under strict computational budgets. Its multi-scale architecture preserves both fine and coarse features throughout the network [48], enabling high-quality predictions even at early stages. MSDNet performs well in both budgeted classification—where a fixed computation budget is shared across a batch—and anytime prediction, surpassing static baselines by allocating minimal resources to easy inputs [22].

3.2. Adaptive Layer Bypassing and Routing

Early exiting follows a unidirectional structure: an input moves forward and exits at the first branch where confidence is sufficient. Adaptive layer skipping offers greater flexibility, enabling non-sequential routing through the network. Rather than committing to a prediction at any intermediate point, the network makes an independent decision at each block: execute or skip. This allows the network to bypass non-consecutive layers and resume computation at any later point, enabling finer control over the computational path [24,25].

Wu et al. [25] propose BlockDrop, which frames block selection as a reinforcement learning problem. A lightweight policy network decides sequentially whether to execute or drop each residual block in a pre-trained ResNet. The reward function encourages accuracy while penalizing executed block count. Applied to ResNet on ImageNet [49], BlockDrop achieves an average speedup of 20% (up to 36% in specific cases) while maintaining 76.4% top-1 accuracy. A significant discovery alongside these experiments was the fact that these learned policies are not random ones. They are subsequently specifically encoded to a set of visual strategies, relying mainly on bypassing some residual blocks with samples presenting prototypical characteristics or simple backgrounds.

Based on these facts, SkipNet [24] proposed a novel methodology in which both a direct gating mechanism inside the network and training it from an end-to-end point of view are implemented. This mechanism allows the elimination of any need for a pre-trained and pre-defined, stable backbone for the network. SkipNet employs a gating function relying on a recurrent neural network (RNN) to observe output feature maps of a layer in question to check the possibility of executing the following layer. In addition to that, and to surmount the main targeted obstacle of non-differentiability for binary bypassing determination, the researchers suggested a mixed learning methodology where supervised learning associated with classification jobs is combined with reinforcement learning approach [24]. This seminal work was able to manifest the capacity to minimize dramatic computation by 30% to 90% based on several benchmarks while maintaining high degrees of accuracy.

By taking adaptive depth to its limits, Convolutional Networks with Adaptive Inference Graphs (ConvNet-AIG) [21] exceeds the simple bypassing approach, heading toward a fully input-dependent computational graph design. Basing their work on the concept of a network with a property to jump to a certain specialized layer the moment it identifies a high-degree concept, ConvNet-AIG researchers have created a graph structure over the layers of this network, built on an input basis. The researchers proved the fact that their model has the capacity to learn various unique inference paths for multiple image categories inside the ImageNet dataset. For instance, pictures of wine bottles were able to skip specialized layers for textured animal fur. The two versions of the ConvNet-AIG model (50-layer and 101-layer) surpass in accuracy levels of their ResNet counterparts

yet only employing 20% and 33% respectively fewer computational operations. This research established the property that dynamic depth inference does not rely mainly on reducing computation operations and costs but is related also to a more intelligent and specialized network architecture [21]. More recently, DynaLay introduces an introspective layer-selection mechanism in which each layer estimates its own utility for a given input and adaptively decides whether to execute, extending input-dependent routing beyond hand-designed gating functions [50].

3.3. Comparative Analysis of Dynamic Depth Methods

The two main approaches associated with dynamic depth inference, adaptive skipping and early exiting, provide a set of different strengths while forming well-suited solutions for various cases treated and discussed. Early exiting, as employed in BranchyNet [23] and MSDNet [22], is characterized mainly by a simple and easier integration and training process, offering natural fitting for anytime predictions, and in parallel, elevated performance gain levels. This approach's main constraint resides in its sequential structure. A single input data must travel through all the layers available to reach its exit point, which may still produce redundant computations in the case where information needed was presented even earlier.

Adaptive layer skipping and routing approaches employed in the likes of SkipNet [24], BlockDrop [25] and ConvNet-AIG [21] offer finer micro-control inside the architecture resulting in a higher chance of higher efficiency levels by permitting non-sequential paths. Nevertheless, this property comes at the price of a more complex network. Training this kind of network will require mainly more sophisticated methods such as reinforcement learning or even hybrid learning in order to control cases of non-differentiable binary decisions, which presents increased instability and higher computational effort compared to standard supervised training models [25,41]. Moreover, routing choices could present overhead, which should be outweighed by the savings from bypassing large layers.

Table 3 compares the principal dynamic depth approaches in terms of their mechanisms, training strategies, advantages, and limitations. The comparison highlights the trade-off between relatively simple early-exit methods and more flexible adaptive skipping or routing approaches, which can provide finer computational control but generally require more complex training and routing mechanisms.

Table 3. Comparative summary of dynamic depth inference strategies.

Method & Citation	Category	Core Mechanism	Training Approach	Key Advantages	Key Limitations
BranchyNet [23]	Early Exiting	Auxiliary classifiers at intermediate layers; confidence thresholding for early termination	Supervised Learning	Simple to implement; ideal for anytime prediction; strong predictable speedups	Sequential processing; potential redundant computation before exit
MSDNet [22]	Early Exiting	Multi-scale architecture with early exits; maintains both coarse and fine features	Supervised Learning	Excellent for budgeted batch classification; high-quality predictions at early stages	Architectural complexity; subject to sequential processing limitation

Table 3. *Cont.*

Method & Citation	Category	Core Mechanism	Training Approach	Key Advantages	Key Limitations
BlockDrop [25]	Adaptive Skipping	RL-based policy network to decide which residual blocks to execute in a pre-trained ResNet	Reinforcement Learning (RL)	Leverages ResNet robustness; learned policies are semantically meaningful	Requires a pre-trained model; RL training can be unstable
SkipNet [24]	Adaptive Skipping	Integrates an RNN-based gating function to decide on skipping the next layer, trained end-to-end	Hybrid Learning (Supervised + RL)	End-to-end training; very high computational reduction (30–90%)	Complex hybrid training; RNN gating overhead
ConvNet-AIG [21]	Adaptive Routing	Learns a full, input-dependent computation graph allowing non-sequential jumps between layers	Supervised (Gumbel–Softmax)	Most flexible pathway; can improve accuracy by specializing pathways	High architectural and training complexity; significant routing overhead

To conclude this section, dynamic depth inference has evolved to be a well-understood set of methods for constructing efficient and flexible deep learning models. By structuring the computational depth on an input level, these networks reached high degrees of the aforementioned efficiency–accuracy trade-off, opening the way for a central evolution of deep learning models in resource-limited environments. In the next section, we will dive similarly into the techniques related to dynamic width inference.

4. Dynamic Width Inference

After exploring the dynamic inference methodologies relying on dynamic depth, which integrates a longitudinal computational modeled path of the model's network, this review paper tackles the concept of dynamic width inference. This set of approaches in dynamic inference modifies the target of the concept itself from various layers inside the network to its internal core, designating the total number of enabled channels or filters inside a singular layer [7,26]. A definition of dynamic width inference could be formulated as follows: the empirical examination proving that not all extracted features by a single convolutional kernel are equally useful and vital in the treatment of every input data processed. The dynamic regulation of the network's width by use of these approaches leads to the construction of a lightweight auxiliary network for simple and easy inputs while leaving the full capabilities of the network for more difficult and complex samples. This methodology offers an optimized stabilization of the efficiency–accuracy trade-off indicator by maintaining a micro-scale network architecture through the continuous elimination of computational redundancies at the core of kernel feature extraction [37,51].

4.1. Dynamic Width Concept

In convolutional neural networks, the width of a layer is defined as the number of its output filters, each specialized in detecting a single type of feature or pattern within its receptive field. The collective output of these filters forms the feature map—the output of the preceding convolutional layer—where the channel axis dimensionality equals the number of defined patterns. On a macro level, the total width of a network defines its representational capacity and computational complexity, typically measured in FLOPs. The dynamic width terminology denotes adaptivity on a channel or kernel level rather than a reduction in active channel count. Channel and filter gating (Section 4.2.1) target active width; dynamic convolution (Section 4.2.2) instead adapts filters weights via input-based attention without altering the active channels count—this is discussed in Section 4.3.

Implementing dynamic inference techniques inside deep learning models significantly diminishes computational redundancy at the feature map level. For example, classifying a simple centered object against a plain background only requires a limited set of filters—detecting edges, colors, or shape—to reach an accurate prediction. Activating additional filters for more complex features is unnecessary in such cases. Nevertheless, in a static network, all filters are computed indiscriminately, incurring an elevated computational cost disproportionate to the task difficulty. Dynamic width methods address this by learning a per-input selection policy: identifying which filters are necessary for a given input and activating only those, reducing the effective layer width dynamically.

4.2. Dynamic Width Inference Architectural Strategies

4.2.1. Filter and Channel Gating

Filter and channel gating is the most direct approach to dynamic width inference, using a lightweight algorithm to select which layer components to activate per input [7,37]. LC-Net by Xia et al. [7] combines a main network (L-Net) with a lightweight control network (C-Net). The C-Net analyzes each input and generates a binary mask determining which L-Net filters are activated; filters assigned zero are bypassed entirely. LC-Net achieves up to 11.9× FLOPs reduction on CIFAR-10 [52] and 1.4× on ImageNet [7]. Figure 2 illustrates the LC-Net filter gating mechanism.

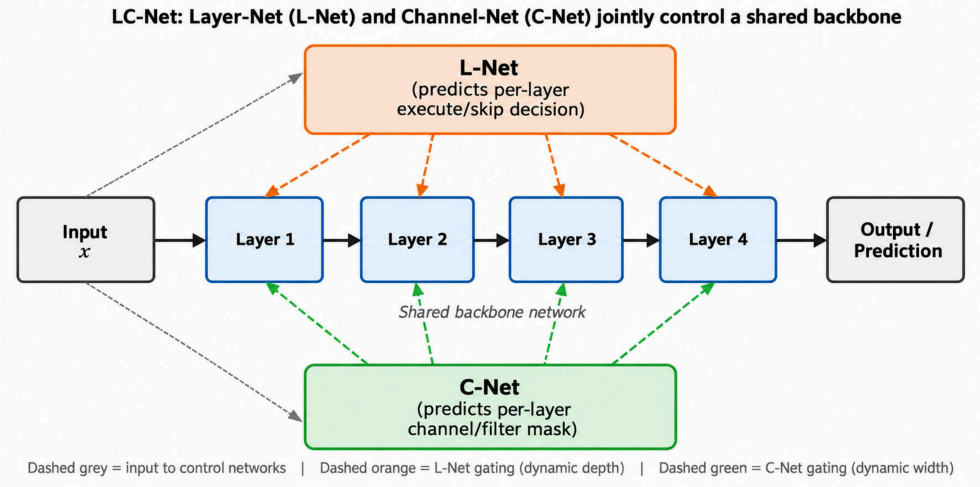


Figure 2. LC-Net dynamic width inference mechanism. The C-Net control network generates a binary mask per input, selectively activating only the necessary filters in the L-Net main network, while crossed channels are skipped [7].

Runtime Neural Pruning [37] applies a similar gating model with pruning decisions made at inference time rather than statically before deployment. End-to-end training of these gating mechanisms typically relies on the Gumbel–Softmax trick [30,31], which provides a differentiable approximation of discrete decisions, allowing gradients to flow through the gating mechanism during backpropagation.

4.2.2. Dynamic Convolutions

Dynamic convolution [26] adopts a distinct strategy from binary gating. Each standard convolutional layer is replaced with N parallel filters; an attention mechanism produces input-dependent weights, and the final convolution output represents a weighted sum of all kernel operations. This effectively yields a custom convolutional filter on a per-input basis.

The major advance of this strategy is enhanced representational capacity without altering the model's structural dimensions. Since the produced filters are aggregated before convolution is performed, the number of FLOPs remains approximately identical to that of a classical single-kernel layer, with only the small overhead of the attention mechanism added. Chen et al. [26] demonstrate that dynamic convolution improves MobileNet accuracy with negligible FLOPs overhead. Contrary to binary gating, dynamic convolution extends representational capacity rather than reducing computation—the two methods target complementary efficiency goals.

4.2.3. Alternative Techniques

Related NAS-based solutions constantly try to reach a single efficient architecture rather than adapting based on treated inputs. MetaPruning [51] trains a meta-network also called PruningNet in order to adequate parameters for candidate pruned structures. Then, it performs an evolutionary search in order to decide which pruned structure to choose under a FLOPs budget constraint. The resulting network is static at deployment, as we include it here as a contrast case. This network shares dynamic width's objective of a channel-level efficiency but do not adapt per input—refer to Section 6.2 for a parallel discussion of OFA [38]. More recent width-adaptive designs such as Slimmable ConvNeXt extend the dynamic-width principle to modern ConvNeXt backbones, allowing a single trained network to operate at multiple width configurations for efficient deployment across heterogeneous devices [53].

4.3. Comparative Critical Analysis

Filter gating strategies such as LC-Net [7] provide direct, channel-wise control over activation sparsity. Dynamic convolution [26] operates at the parameter level, tuning filter weights through a differentiable attention mechanism—a more stable approach from a training perspective. Filter gating networks introduce a gating overhead that must be offset by the savings from bypassed filters. Dynamic convolution's overhead is limited to the small attention model. The two methods serve opposite objectives: filter gating reduces computation by deactivating channels, while dynamic convolution increases capacity by aggregating filters. Binary gating strategies face non-differentiability, addressed through reinforcement learning or the Gumbel–Softmax approximation. Dynamic convolution's differentiable attention mechanism supports standard end-to-end backpropagation, making training more stable and reproducible [26].

LC-Net [7] achieves large FLOPs reductions on CIFAR-10 while maintaining ImageNet accuracy. Dynamic convolution [26] improves MobileNetV2 Top-1 accuracy by 2.5% on ImageNet with a 4% FLOPs increase—trading a small efficiency cost for a meaningful accuracy gain. A persistent challenge across dynamic width methods is the gap between FLOPs reduction and actual latency improvement. Per-input channel masking produces irregular memory access patterns that GPUs—optimized for dense, uniform tensor operations—cannot efficiently parallelize. Unlike static pruning, which removes entire channels and produces structured sparsity, dynamic channel masking generates irregular, input-specific sparsity that requires custom hardware support for practical acceleration [7,37].

The difference between theoretical FLOPs counts and measured latency is documented outside dynamic inference. Latency on GPUs does not scale linearly with FLOPs count: convolutional layers with a with the same number of FLOPs can offer different measured latency readings-based channel count and feature-map dimension. This is due to the GPU throughput being staircase-shape with respect to the dimension of the channel as well as memory-bound operations not benefiting from FLOPs count minimization. For instance, the ResNet-50 presents the same count of FLOPs for both the 7×7 convolution and the

3 × 3 bottleneck convolution. However, the former performs roughly 60% slower on chip, as shown in Shen et al. [54]. In the case of depth-wise convolution, a reduced number of FLOPs is exhibited, with respect to dense convolutions but comparable to GPU latency. This is caused by memory-bandwidth-bound rather than compute-bound. This directly affects dynamic inference: per-input channel masking (Section 4) and spatial sparsity (Section 5) both produce irregular, input-dependent memory access patterns that classical GPUs—optimized for dense, uniform tensor operations—cannot efficiently parallelize, which is why several works in this review report large FLOPs reductions alongside comparatively modest measured speedups [7,30].

Dynamic width inference implements conditional computation at the intra-layer level, enabling fine-grained efficiency control that complements dynamic depth approaches. Frameworks such as DDI [8] and OFA [38] demonstrate that combining dynamism across depth, width, and resolution yields compounded gains exceeding any single dimension alone.

5. Dynamic Spatial Resolution

This section examines the third dimension of dynamic inference: spatial resolution. The main concept behind dynamic spatial resolution inference lies in the fact that the informational value within a given sample does not present a uniform distribution over its spatial components [17,18,31]. Taking any image as an example, the semantic core needed to achieve a classification task usually resides in specific areas, while vast segments of the background offer minimal value to the final network output. Dynamic spatial resolution inference approaches revolve around exploiting this spatial redundancy through adaptive accuracy selection at which a given input sample—or even a computational layer feature map—is processed. The main motivation of such an approach is to focus the majority of computational resources on the most informative image segments while reducing effort on less important ones, thus reaching dramatically increased efficiency levels without degrading accuracy [17,30,55].

5.1. Spatial Adaptivity Framework

Spatial adaptivity defines a core concept in human natural perception, in which the fovea offers high-resolution representation for the focus of attention, while peripheral vision operates at lower resolution [56]. Translating this concept to convolutional neural networks, strategies integrating dynamic spatial resolution depart from static, fixed high-resolution treatment conventions applied uniformly to every sample. This approach proposes basing the spatial fidelity of a given input on its computational difficulty.

Two complementary motivations underlie these strategies. The first is that high-resolution images of easily detectable objects contain a vast majority of redundant pixels. Processing these regions at lower resolutions accurately reflects their actual contribution to the final prediction, yielding direct savings in computational effort and complexity—particularly in early layers, which scale spatial dimensions significantly [17,29]. The second motivation concerns the resulting feature maps of convolutional layers, where activation is typically sparse. Only a specific subset of spatial regions exhibits high activation responses. Applying convolution uniformly across the entire feature map—including low-activation segments—incurs significant and unnecessary computational cost [18,30].

5.2. Architectural Design Approaches

5.2.1. Input Resolution Scaling Strategies

Input scaling strategies decide, at early stages of the inference pipeline, the resolution level at which a given input sample will be processed throughout the entire network. The

feature map size is directly proportional to the input size, reducing FLOPs quadratically with the resolution reduction factor. These strategies operate at a global level, affecting the entire computational pipeline from the input stage [17,29].

Resolution Adaptive Network (RANet) [17] is a key seminal work in this category. This architecture handles each input progressively, from coarse to fine resolution, using a sequence of auxiliary networks operating at increasing resolutions. A treated input is first handled by a lightweight low-resolution network; if the confidence prediction from the classifier's treatment exceeds a given threshold, the sample exits immediately. Otherwise, it is passed to a second, high-resolution network, which uses the coarse features previously extracted. Therefore, only 'hard' samples reach the full-resolution path. Less complex samples are directed through lower-resolution computational routes, while more complex inputs are processed at full resolution. Early resolution decision-making allows the network to exploit the fact that sample difficulty can often be assessed from its low-frequency content. RANet demonstrates an average 40% reduction in computation on ImageNet while maintaining top-1 accuracy [17]. Figure 3 illustrates the RANet multi-resolution routing mechanism.

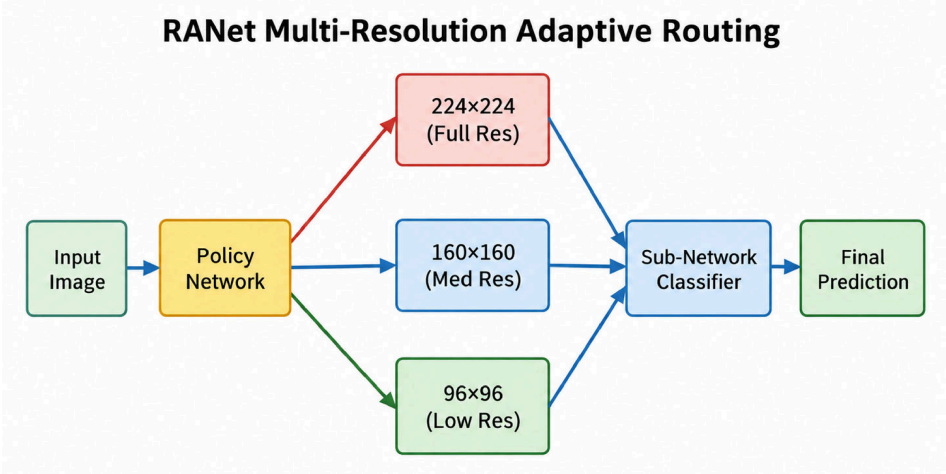


Figure 3. RANet resolution adaptive routing. A policy network estimates input complexity and routes each sample to the appropriate resolution pathway, achieving up to 40% FLOPs reduction while preserving classification accuracy [17].

The Adaptive Resolution Network (AR-Net) [29] extends this specific feature of RANet to sequential data, addressing video recognition tasks efficiently. This network dynamically decides the resolution at which each selected frame from a video clip is processed. It identifies that each frame in a video clip holds a unique informative value; thus, redundant frames can be processed at lower resolutions without compromising recognition accuracy, achieving significant savings at the inference level [29]. AR-Net represents an important step in demonstrating that dynamic spatial resolution inference is applicable beyond still-image classification to temporally structured data, with adaptive resolution selection operating across both the spatial and temporal dimensions of the video stream.

5.2.2. Sparsity and Adaptive Computation

Instead of making a single global decision at the input level, a more fine-grained strategy resides in adaptively computing features only at specific informative regions within intermediate network layers, primarily reducing redundancy at the feature map level [18,30,31].

Verelst et al. [30] present in their seminal work a spatial gating framework operating within specific network blocks. A lightweight activation branch determines, for each spatial region in a given feature map, whether to apply a heavy convolutional operation or to bypass it entirely, allowing the feature map to pass unchanged. This process yields a spatially sparse map in which convolutions are strictly applied only at informative locations. This strategy achieves a significant inference speedup in human pose estimation on standard benchmark datasets [30].

Xie et al. [31] propose in their Spatially Adaptive Inference with Stochastic Feature Sampling and Interpolation work a technique in which the network stochastically samples features at selected spatial regions based on their activation values. Features at unsampled locations are interpolated from the previously sampled ones, substituting the cost of a full convolutional pass with a cheaper sampling and interpolation mechanism, achieving optimized FLOPs on both classification and segmentation tasks [31]. This stochastic sampling process also serves as a regularizer during the training phase.

Spatially Adaptive Computation Time (SACT) [18] represents another major seminal work in dynamic spatial resolution. This approach introduces a halting score at the spatial location level of a ResNet block feature map. The network applies additional computational steps at locations where the halting score has not yet reached a predefined threshold, creating an effective dynamic depth that varies by spatial region. SACT represents one of the first works combining dynamic depth and spatial adaptivity [18,57].

5.3. Comparative Analysis

Different characteristics and properties emerge from a comparative critical analysis of the techniques discussed in this section. Input scaling strategies such as RANet [17] make coarse, network-level decisions that affect the entire computational pipeline. In contrast, sparsity and spatial adaptivity methods such as [30,31] offer finer and more controlled mechanisms, selecting computation on a spatial region basis. RANet represents a lightweight solution, with its routing overhead easily outweighed by the savings from processing simple inputs at lower resolutions. Gating mechanisms in sparsity strategies incur higher overhead, which must be balanced by the large number of bypassed operations [30].

Input scaling networks offer more practical and universally applicable solutions across architectures. In contrast, spatial adaptivity solutions are powerful and well-suited to tasks with naturally localized informative regions, such as segmentation and pose estimation. Implementing input scaling strategies is straightforward, requiring only simple routing decisions. However, efficient implementation of sparsity solutions requires handling irregular memory access patterns that cannot be efficiently accelerated on standard hardware platforms in most cases.

Table 4 compares the main dynamic spatial resolution strategies according to their mechanisms, granularity, advantages, and challenges. The comparison shows that input-resolution scaling provides a relatively simple global adaptation strategy, whereas spatially fine-grained methods can target computation more selectively but introduce greater control-flow and hardware-efficiency challenges.

Table 4. Comparative summary of dynamic spatial resolution strategies.

Approach	Core Mechanism	Seminal Work(s)	Granularity	Key Advantage	Key Challenge
Input Resolution Scaling	Dynamically selects the resolution for the entire input image/video	RANet [17], AR-Net [29]	Coarse (Global)	Massive FLOPs reduction; easy to implement	May lose fine details; single decision point
Spatial Feature Sparsity	Learns to skip convolution at less informative spatial locations in feature maps	Dyn. Conv [30]	Fine (Local)	High accuracy; excellent for sparse tasks like pose estimation	High decision overhead; irregular computation hard to accelerate
Stochastic Spatial Sampling	Probabilistically samples features at sparse locations and interpolates the rest	Spatial Adapt. [31]	Fine (Local)	Replaces conv cost with cheap interpolation; acts as regularizer	Stochasticity complicates training and inference reproducibility
Spatially Adaptive Depth	Varies computational steps per spatial location using a halting score	SACT [18]	Fine (Local)	Combines dynamic depth and spatial adaptivity; pioneering multi-dimensional approach	Complex training; significant control-flow overhead

5.4. Training Obstacles and Recommendations

Integrating dynamic spatial resolution inference solutions comes with several training challenges. These challenges are largely shared with dynamic width inference techniques, targeting mainly the problem of non-differentiability in decision making [30,31,41].

1. Confidence Thresholding with Supervised Training: Similar to early-exit depth solutions (Section 3.1), RANet [17] trains its sequence of resolution-specific auxiliary networks with a standard multi-exit supervised loss, using softmax confidence at inference. This strategy grants the model the decision whether a sample proceeds to a higher-resolution stage or is removed from its path.
2. Gumbel–Softmax Relaxation: The Gumbel–Softmax technique [30,31] remains a valid strategy for discrete sampling, offering end-to-end differentiability and allowing gradients to flow freely through the sampling mechanism.
3. Auxiliary Loss Functions: Auxiliary losses integrated in methods such as SACT [18] encourage early halting, directly shaping the spatial computational map as part of the training objective.

Applications of dynamic spatial resolution have achieved compelling results. Networks such as RANet [17] and AR-Net [29] demonstrate that dynamic spatial resolution selection achieves 30% to 50% reduction in computational effort while maintaining accuracy. Sparsity and spatial adaptivity methods such as [30] present significant speedups of 1.5× to 2× on practical hardware platforms for predefined tasks.

Dynamic spatial resolution represents the third cornerstone of dynamic inference, targeting spatial redundancy to achieve efficiency gains that complement those from depth and width adaptation. This approach offers a well-structured set of solutions to eliminate redundancy from data grids, either through input scaling techniques or feature-based sparsity mechanisms. Applying these strategies allows considerable computational savings

while also facing hardware adaptation limitations in high-granularity techniques. Most prominent approaches combine multiple dimensions, as demonstrated in DDI [8]. The following section concentrates on these multi-dimensional strategies to demonstrate their efficiency and flexibility.

6. Multi-Dimensional Hybrid Dynamic Methods

The three preceding sections have examined dynamic depth, width, and spatial resolution independently. Each targets a distinct form of redundancy, and none is mutually exclusive. Under the reported experimental conditions, combining these dimensions has demonstrated complementary or compounded computational savings [7,8,17,24,38]. Based on this, several approaches have targeted more sophisticated techniques known as multi-dimensional hybrid dynamic networks. These strategies overcome the constraints of single-axis solutions by combining computational processes across two or more axes simultaneously, reaching a more structured control flow while maintaining a stable accuracy–efficiency trade-off [39,58].

6.1. Collaborative Requirements of Multi-Dimensional Dynamism

Multi-dimensional dynamic inference targets the redundancies defined across all three of its architectural axes. For instance, a network relying solely on dynamic depth will still execute all available channels in every active layer, bypassing filter-level redundancy [7]. Similarly, a network employing only dynamic width will propagate features through all layers, ignoring any potential early exiting opportunity [21,24]. Finally, a network based exclusively on dynamic spatial resolution will reduce the data size processed but will still apply a uniform depth and width model to the modified input sample [17,18].

Collaboration between these three distinct axes becomes essential when considering a given sample. A sample classified as easy may present shallow features addressable by dynamic depth, require only a minimal set of filters (dynamic width), and be accurately processed at lower resolution (dynamic spatial resolution). Combining these approaches aggregates computational savings: bypassing layers reduces depth-based FLOPs, pruning channels reduces width-based FLOPs, and resolution scaling reduces spatial FLOPs—yielding a compounded total saving that dramatically exceeds any single-method result [8,39]. Nevertheless, in that case, even if the total saving exceeds the single-axis results, the amount of the gain still depends on the model, the task assigned to it and the deployment environment. In addition to that, multi-axis optimization may exhibit an offset in its theoretical gain due to the additional complexity it involves.

6.2. Architectural Solutions for Hybridization

Several seminal works have proposed architectural solutions to maximize the efficiency and accuracy of integrating such techniques. A pioneering work in this area is the Dual Dynamic Inference (DDI) framework [8]. This framework incorporates multi-dimensional dynamic inference across depth and width simultaneously via a Multi-Grained Dynamic Inference mechanism, which learns two complementary policies. The first, a layer-wise policy, determines whether to bypass an entire network block (dynamic depth), while a channel-wise policy, applied between active blocks, selects which channels to activate (dynamic width) [8]. These decisions are jointly orchestrated through regularization. The results of implementing DDI are compelling, demonstrating that combining multiple dynamic axes yields considerably greater efficiency than any single dynamic dimension alone. Figure 4 provides a conceptual overview of the DDI multi-dimensional orchestration framework.

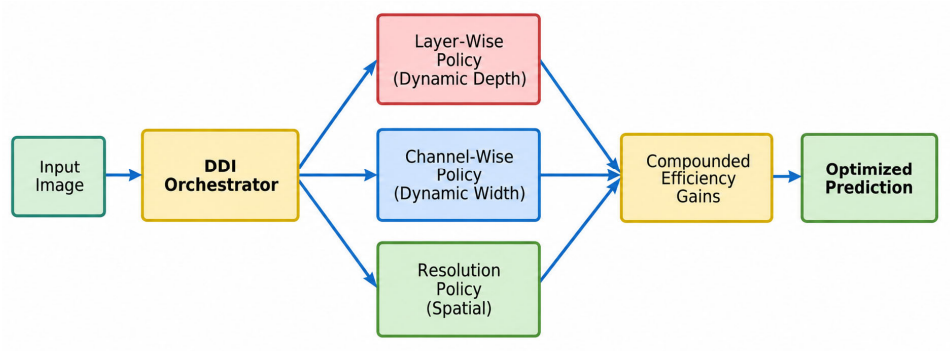


Figure 4. DDI multi-dimensional dynamic inference framework. A central orchestrator jointly optimizes depth and width policies, yielding compounded efficiency gains that, under the reported experimental conditions, can surpass those of single-dimensional approaches [8].

Another impactful seminal work in this field is the Once-for-All (OFA) network [38]. OFA presents a well-structured supernet that encapsulates multi-level flexibility. This model is trained once as a large network embodying a wide set of lightweight auxiliary subnetworks within its structure, spanning a broad range of depth, width, kernel size, and resolution parameters. After training, OFA can be specialized by extracting specific subnetworks that meet the resource constraints of the target deployment device. Despite the computational path being determined post-training, OFA's architecture and training primarily address the challenge of managing a continuously multi-level trade-off space [38]. OFA's auxiliary network selection is determined only at deployment time. This is done by means of a one-time architecture search under a threshold of resources, not on an input-basis at inference. A specialized auxiliary network, once extracted, is applied uniformly to all treated samples, static in that case. OFA is included in this work as a boundary case, showcasing the variance between fully static architecture search and purely input-based dynamic inference—see also the discussion of MetaPruning in Section 4.2.3.

Recent works have pushed toward fully adaptive, multi-dimensional dynamic inference networks that couple spatial resolution with adaptive depth. The Glance and Focus Network (GFNet) [58] first examines a low-resolution version of the input and then iteratively focuses on a sequence of small, high-resolution patches selected by a reinforcement-learning policy. Because the process terminates as soon as the model is sufficiently confident, resolution adaptation and adaptive computation depth are optimized jointly rather than in isolation: computational resources are allocated along both the spatial and the depth dimensions, and full-resolution processing is reserved only for the regions and the samples that require it. GFNet is backbone-agnostic and reduces the measured on-device latency of MobileNet-V3 by roughly 20% without accuracy loss, illustrating that multi-dimensional dynamism can translate into real hardware gains rather than FLOPs reductions alone [58,59].

Furthermore, the Dynamic Slimmable Network (DS-Net) [39] addresses a constraint that limits most of the width-based methods discussed in Section 4: the mismatch between theoretical and realized savings. Instead of masking individual channels, DS-Net predicts a slimming ratio per input through a double-headed dynamic gate and then slices a contiguous portion of each layer's filters. Because the retained parameters remain stored contiguously in memory, the resulting sub-network is dense and can be executed efficiently on standard hardware, avoiding the irregular access patterns that undermine masking-based gating. DS-Net is trained in two disentangled stages—supernet training followed by gate training—which stabilizes an optimization problem that is otherwise prone to collapse. It therefore demonstrates that the orchestration problem in multi-dimensional dynamism is not purely algorithmic: the form the dynamism takes determines whether the predicted

savings survive contact with the hardware [39]. Boosted Dynamic Networks pursue a related but distinct strategy, applying a boosting-inspired training scheme in which a sequence of lightweight predictors is progressively refined by deeper, more expressive ones, combining ensemble-learning principles with early-exit-style dynamic computation [60].

6.3. Challenges of Training and Orchestration

The improved capabilities of multi-dimensional dynamic inference networks come at the cost of increased training difficulty, complexity, and instability.

1. Policy Interconnection and Conflicts: Multi-dimensional networks maintain a set of interconnected policies for each dynamic dimension [8]. For instance, aggressively pruning channels in a given layer may degrade its output quality, weakening the confidence of subsequent early exit decisions. Orchestrating the relationships between these policies and their discrete outputs to yield an optimal inference pathway constitutes a non-trivial optimization problem.
2. Non-Differentiability: The training pipeline must backpropagate through multiple parallel paths of discrete decisions. Techniques such as reinforcement learning and Gumbel–Softmax relaxation can be applied to address this issue [30,31]. However, the explosion of possible decision states introduces complex optimization landscapes prone to instability.
3. Balanced Optimization: The loss function must balance the savings targeted across each dynamic dimension. An inadequately weighted objective may cause the model to over-optimize along one dynamic axis at the expense of another, preventing the intended synergistic effect [8,39].
4. Gating and Routing Cost: Auxiliary components promoting dynamic inference, such as gating networks (Section 4), policy networks (Sections 3 and 5), are accompanied with their respective computational and weights cost. As shown in both Sections 4.3 and 5.3, this cost shall be outweighed by bypassed computation savings, in order to reach a clean efficiency gain. However, in the case of multi-dimensional architectures, this cost compounds across all active policy component. In this case, the net cost-to-saving ratio is often considered an under-reported design consideration.

In comparing DS-Net's [39] disentangled stage training against DDI's [8] joint regularization solution, it is found that DS-Net takes longer time for pipeline training but reaches a more stable optimization landscape, while DDI's joint approach trains the pipeline faster but in a more sensitive way to the loss-weighting between axes. Therefore, disentangled training comes more favorable when engineering time is available and stability becomes more fundamental than training-time cost.

To address these challenges, researchers have explored tailored advanced training strategies. Some integrate progressive hardening, in which the network is first trained to be dynamic along one dimension before the second dynamic dimension is enabled [38]. Other schemes decouple the search for a computational path from the training of the weights, drawing on one-shot and reinforcement-learning-based architecture search, in which separate policy or search components govern each dimension while sharing a global objective [55,61,62]. A growing interest in latency-aware optimization has emphasized the direct incorporation of hardware latency measurements into the training loss.

Multi-dimensional hybrid networks represent the most complete realization of conditional computation. DDI [8], DS-Net [39], and OFA [38] demonstrate that intelligently orchestrating dynamism across depth, width, and resolution can yield efficiency gains that, under the reported experimental conditions, exceed those of single-dimension methods. These frameworks transform the network from a static pipeline into an input-responsive adaptive system.

Table 5 provides an overall comparison of representative dynamic inference methods across the categories discussed in this review, including their datasets, backbones, reported efficiency metrics, hardware context, and verification status. The table highlights the diversity of evaluation settings across the literature and therefore provides a consolidated reference for interpreting the reported efficiency gains rather than a direct ranking of the methods.

Table 5. Comparative summary of representative dynamic inference methods, reported efficiency results, hardware context, and verification status.

Method	Dataset	Backbone	Hardware (Where Reported)	Metric	Result	Verification Status
BranchyNet [23]	MNIST	AlexNet variant	Not reported	Exit rate/FLOPs reduction	>75% exit at branch 2, >50% computation reduction	Verified against [23]
BlockDrop [25]	ImageNet	ResNet-101	Not reported	FLOPs reduction	20% avg, up to 36%, 76.4% top-1	Verified against [25]
SkipNet [24]	ImageNet	ResNet-101	Not reported	FLOPs reduction	30% (ImageNet); 30–90% across 4 datasets	Verified against [24]
LC-Net [7]	CIFAR-10/ImageNet	ResNet + L-Net + C-Net	Not reported	FLOPs reduction	11.9×/1.4×	Verified against [7]
Dynamic Convolution [26]	ImageNet	MobileNetV2	Not reported	Accuracy/FLOPs trade	+2.5% top-1, +4% FLOPs	Verified against [26]
RANet [17]	ImageNet	Multi-scale cascade	Not reported	FLOPs reduction	30–50% combined with AR-Net (§5.4)	As originally reported [17]
GFNet [58]	—	MobileNet-V3	Mobile CPU	Measured latency	~20%	As originally reported [58]
Verelst et al. [30]	ImageNet (detection)	Various	Not reported	FLOPs reduction	33% (originally reported; not independently verified in this revision)	As originally reported [30]
MetaPruning [51]	—	—	N/A	—	N/A—not per-input dynamic	N/A (not per-input dynamic)

7. Open Challenges and Future Directions

Despite major advances, several obstacles continue to limit the practical implementation of dynamic inference strategies, motivating the following future research directions.

Narrowed Hardware Optimization Gap. One of the most critical problems lies in the detachment between theoretical FLOPs reduction and actual latency improvement on hardware platforms. Irregular computational patterns, such as those arising from channel pruning [7] or spatial sparsity [30], can produce poor parallelism and inefficient memory access on classical GPUs, negating the conceptual savings [63,64]. Future work must target the co-design of dynamic inference methods with specialized hardware platforms. Pursuing structured sparsity aligned with data access patterns, or developing dynamic inference models that inherently produce denser tensors—such as input resolution scaling [17]—represents a prominent direction. Incorporating actual hardware latency into the training loss, as demonstrated in several NAS studies [65], offers another promising pathway [66]. Future research must aim at co-designing dynamic inference solutions by means of specialized hardware platforms, such as NPUs [67], as well as edge AI accelerators. These technologies differ substantially in their execution from general-purpose GPUs and may favor structured dynamism over irregular per-channel or per-input masking.

Powerful and Scalable Training Approaches. Training dynamic inference models, especially multi-dimensional instances, remains a difficult and unstable process. The non-differentiability of discrete decisions is partially addressed through RL approaches [25] and Gumbel–Softmax relaxation [30,31]. However, these solutions remain sensitive to hyperparameters and scale poorly with growing decision spaces. Future work must develop more stable end-to-end training strategies. Differentiable neural architecture search (DNAS) [68] represents a potential pathway for learning dynamic routing policies [69]. Furthermore, gradient conflicts in multi-exit models [41] and policy interdependencies in hybrid networks [8] demand carefully designed objectives that ensure orchestrated collaboration across all dynamic dimensions.

Expanded Application Areas. Most dynamic inference works have targeted image classification. Future research must extend dynamic inference methods to additional domains [70–72]. Natural language processing, with its sequential structure and transformer layers [73], and reinforcement learning represent promising targets [74] for complexity-aware dynamic adaptation [75,76]. Integrating dynamic inference in video recognition [29,77] and medical imaging, where large data volumes and spatially localized informative regions are prevalent, holds significant potential [78,79]. CNN-based methods are also increasingly used in engineering applications, including structural health monitoring and damage detection [80]. Input-adaptive dynamic networks have also begun to target object detection under resource-constrained deployment, indicating that the depth- and width-adaptive mechanisms surveyed throughout this review extend naturally beyond classification tasks [81].

Theoretical Robustness. Although the practical results of dynamic inference are compelling, its theoretical foundations remain limited. Core open questions concern the development of worst-case performance guarantees. Evidence also suggests that adaptive models are vulnerable to adversarial examples [82] specifically crafted to defeat their efficiency mechanisms, forcing worst-case computation rather than misclassification [83,84]. Future work into the verification and adversarial robustness of dynamic inference solutions represents a crucial step toward safety-critical deployment.

Practical Implementation. To achieve wide adoption, dynamic inference approaches must overcome practical integration barriers. Developing standardized software packages alongside suitable compiler mechanisms for dynamic graphs barely supported in deep learning frameworks [85,86], is a must. Moreover, building global benchmarks that test dynamic inference strategies not just on their accuracy and FLOPs levels but also on real-world indicators [87] is an essential area of development. Power consumption, throughput, and latency under varied load conditions are essential indicators for fair comparison and practical evaluation [8]. In our future work, we identify the need for an accompanying open-source script, which will cover basically FLOPs/weights metrics and basic simulation for the architectures discussed in this work. QoS-aware inference frameworks that dynamically adjust network depth to meet latency service-level objectives illustrate a concrete step in this direction [88].

Figure 5 provides a visual comparison of FLOPs reductions relative to the corresponding static baselines for selected representative methods, grouped according to dataset and backbone. The values shown in the figure were independently checked against the original publications. Because the methods differ in terms of datasets, backbone architectures, and training regimes, the bars should not be interpreted as a direct cross-method ranking. Other methods discussed in this review are not included in Figure 5 either because their reported metric is not directly comparable to the FLOPs-reduction values plotted in the figure or because the corresponding values could not be independently verified from the original publications. Their reported results and verification status are summarized in Table 5.

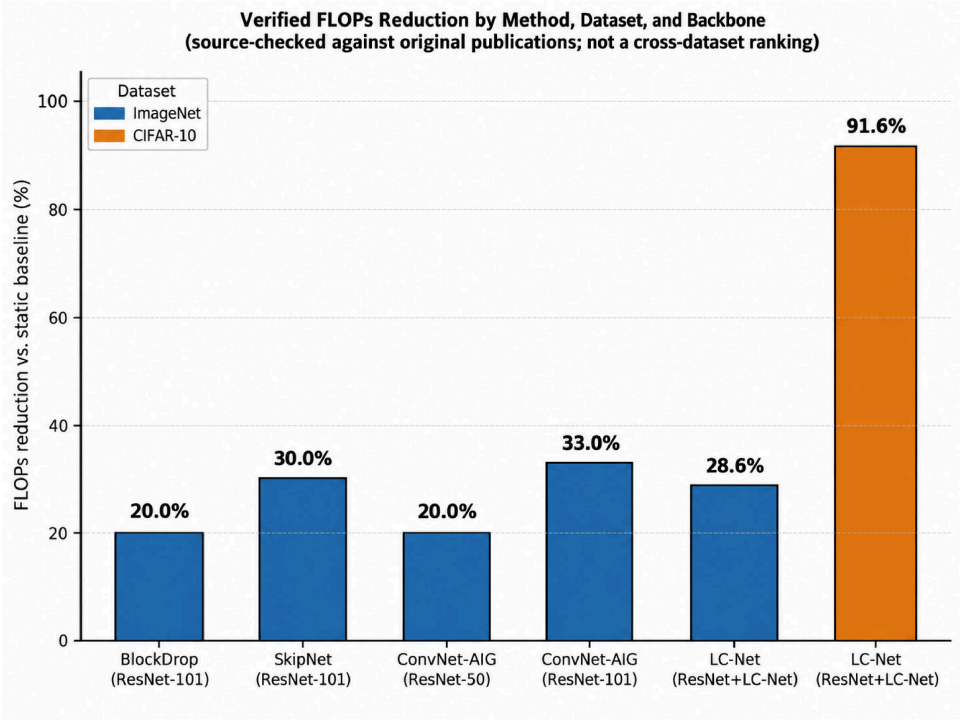


Figure 5. Verified FLOPs reduction relative to static baselines for selected representative methods discussed in this review: BlockDrop [25], SkipNet [24], ConvNet-AIG [21], and LC-Net [7].

8. Conclusions

This review shows that the main challenge in dynamic inference is no longer simply reducing theoretical computation, but ensuring that adaptive computation translates into measurable efficiency on real hardware. Across the methods examined, reductions in FLOPs do not always lead to proportional improvements in latency because routing overhead, irregular memory access, hardware utilization, and implementation constraints can significantly affect realized performance. This gap is especially important for highly dynamic methods, where the cost of making routing or gating decisions can partially offset the computational savings they are designed to achieve.

Another important takeaway is that greater adaptivity generally comes with greater optimization complexity. Multi-dimensional approaches can combine complementary sources of efficiency and provide additional gains beyond single-axis methods, but they also introduce stronger interactions between routing policies, training objectives, and deployment constraints. As a result, the most effective dynamic inference system is not necessarily the one with the largest theoretical FLOPs reduction, but the one that achieves a favorable balance among computational savings, accuracy preservation, training stability, and hardware efficiency.

Several challenges therefore remain important for future research. These include narrowing the gap between theoretical FLOPs reduction and practical latency, improving the stability of training strategies involving discrete routing decisions, extending dynamic inference beyond image classification to domains such as video, NLP, and medical imaging, and developing stronger robustness guarantees for adaptive models. Progress in these areas will increasingly require closer co-design between network architecture, training strategy, compiler support, and target hardware.

Overall, the transition from static to input-adaptive inference represents an important direction toward more resource-aware deep learning systems. The next stage of development should focus not only on how much computation can be skipped, but also on whether

these savings can be realized consistently, efficiently, and reliably in real-world deployment environments.

Author Contributions: Conceptualization, J.K.; methodology, J.K.; investigation, J.K.; writing—original draft preparation, J.K.; writing—review and editing, R.I.; supervision, R.I.; project administration, R.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: During the preparation of this manuscript, the authors used Claude Sonnet 4.5 (Anthropic) solely for English language editing and grammatical correction. All suggestions were reviewed and approved by the authors, who take full responsibility for the final content of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AIG	Adaptive Inference Graph
AR-Net	Adaptive Resolution Network
CNN	Convolutional Neural Network
DDI	Dual Dynamic Inference
DNAS	Differentiable Neural Architecture Search
FLOPs	Floating-Point Operations
IoT	Internet of Things
MAC	Multiply–Accumulate
mAP	mean Average Precision
MDP	Markov Decision Process
MSDNet	Multi-Scale Dense Network
NAS	Neural Architecture Search
NLP	Natural Language Processing
OFA	Once-for-All
RANet	Resolution Adaptive Network
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SACT	Spatially Adaptive Computation Time

References

1. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
2. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
3. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
4. Purwono, P.; Ma'arif, A.; Rahmaniar, W.; Fathurrahman, H.I.K.; Frisky, A.Z.K.; Haq, Q.M.U. Understanding of Convolutional Neural Network (CNN): A Review. *Int. J. Robot. Control Syst.* **2022**, *2*, 739–748. [CrossRef]
5. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]

6. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
7. Xia, W.; Yin, H.; Dai, X.; Jha, N.K. Fully Dynamic Inference with Deep Neural Networks. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 962–972. [CrossRef]
8. Wang, Y.; Shen, J.; Hu, T.-K.; Xu, P.; Nguyen, T.; Baraniuk, R.; Wang, Z.; Lin, Y. Dual Dynamic Inference: Enabling More Efficient, Adaptive, and Controllable Deep Inference. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 623–633. [CrossRef]
9. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
10. Menghani, G. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Comput. Surv.* **2023**, *55*, 259. [CrossRef]
11. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning Both Weights and Connections for Efficient Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 7–12 December 2015; pp. 1135–1143.
12. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 2704–2713. [CrossRef]
13. Denton, E.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 8–13 December 2014; pp. 1269–1277. Available online: https://arxiv.org/abs/1404.0736 (accessed on 29 September 2026).
14. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges. *IEEE Signal Process. Mag.* **2018**, *35*, 126–136. [CrossRef]
15. Cheng, H.; Zhang, M.; Shi, J.Q. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 10558–10578. [CrossRef] [PubMed]
16. Kuzmin, A.; Nagel, M.; van Baalen, M.; Behboodi, A.; Blankevoort, T. Pruning vs. Quantization: Which Is Better? In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10–16 December 2023; Volume 36, pp. 62414–62427. [CrossRef]
17. Yang, L.; Han, Y.; Chen, X.; Song, S.; Dai, J.; Huang, G. Resolution Adaptive Networks for Efficient Inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 2369–2378.
18. Figurnov, M.; Collins, M.D.; Zhu, Y.; Zhang, L.; Huang, J.; Vetrov, D.; Salakhutdinov, R. Spatially Adaptive Computation Time for Residual Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1039–1048.
19. Bengio, Y.; Léonard, N.; Courville, A. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv* **2013**, arXiv:1308.3432.
20. Han, Y.; Huang, G.; Song, S.; Yang, L.; Wang, H.; Wang, Y. Dynamic Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 7436–7456. [CrossRef] [PubMed]
21. Veit, A.; Belongie, S. Convolutional Networks with Adaptive Inference Graphs. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–18.
22. Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; Weinberger, K.Q. Multi-Scale Dense Networks for Resource Efficient Image Classification. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
23. Teerapittayanon, S.; McDanel, B.; Kung, H.T. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In Proceedings of the International Conference on Pattern Recognition (ICPR), Cancún, Mexico, 4–8 December 2016; pp. 2464–2469. [CrossRef]
24. Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; Gonzalez, J.E. SkipNet: Learning Dynamic Routing in Convolutional Networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 409–424.
25. Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L.S.; Grauman, K.; Feris, R. BlockDrop: Dynamic Inference Paths in Residual Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8817–8826.
26. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic Convolution: Attention over Convolution Filters. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11030–11039.
27. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
28. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-Local Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7794–7803.

29. Meng, Y.; Lin, C.-C.; Panda, R.; Sattigeri, P.; Karlinsky, L.; Oliva, A.; Saenko, K.; Feris, R. AR-Net: Adaptive Frame Resolution for Efficient Action Recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 86–104.
30. Verelst, T.; Tuytelaars, T. Dynamic Convolutions: Exploiting Spatial Sparsity for Faster Inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 2320–2329.
31. Xie, Z.; Zhang, Z.; Zhu, X.; Huang, G.; Lin, S. Spatially Adaptive Inference with Stochastic Feature Sampling and Interpolation. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 531–547.
32. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
33. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520. [CrossRef]
34. Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324. [CrossRef]
35. Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Deveci, M.; Parmar, M. A Review of Convolutional Neural Networks in Computer Vision. *Artif. Intell. Rev.* **2024**, *57*, 99. [CrossRef]
36. Bhatt, D.; Patel, C.; Talsania, H.; Patel, J.; Vaghela, R.; Pandya, S.; Modi, K.; Ghayvat, H. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics* **2021**, *10*, 2470. [CrossRef]
37. Lin, J.; Rao, Y.; Lu, J.; Zhou, J. Runtime Neural Pruning. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 2181–2191.
38. Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; Han, S. Once-for-All: Train One Network and Specialize It for Efficient Deployment. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
39. Li, C.; Wang, G.; Wang, B.; Liang, X.; Li, Z.; Chang, X. Dynamic Slimmable Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 8607–8617.
40. Jang, E.; Gu, S.; Poole, B. Categorical Reparameterization with Gumbel-Softmax. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
41. Li, H.; Zhang, H.; Qi, X.; Yang, R.; Huang, G. Improved Techniques for Training Adaptive Deep Networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1891–1900.
42. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
43. Gong, C.; Chen, Y.; Luo, Q.; Lu, Y.; Li, T.; Zhang, Y.; Sun, Y.; Zhang, L. Deep Feature Surgery: Towards Accurate and Efficient Multi-Exit Networks. In *Computer Vision–ECCV 2024*; Springer: Cham, Switzerland, 2024; pp. 435–451. [CrossRef]
44. Li, Z.; Xu, P.; Chang, X.; Yang, L.; Zhang, Y.; Yao, L.; Chen, X. When Object Detection Meets Knowledge Distillation: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10555–10579. [CrossRef] [PubMed]
45. Rahmath, P.H.; Srivastava, V.; Chaurasia, K.; Pacheco, R.G.; Couto, R.S. Early-Exit Deep Neural Network—A Comprehensive Survey. *ACM Comput. Surv.* **2024**, *57*, 75. [CrossRef]
46. Panda, P.; Sengupta, A.; Roy, K. Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition. In Proceedings of the Design, Automation & Test in Europe Conference (DATE), Dresden, Germany, 14–18 March 2016; pp. 475–480.
47. Regol, F.; Chataoui, J.; Coates, M. Jointly-Learned Exit and Inference for a Dynamic Neural Network: JEI-DNN. *arXiv* **2023**, arXiv:2310.09163.
48. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708. [CrossRef]
49. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
50. Mathur, M.; Plis, S. DynaLay: An Introspective Approach to Dynamic Layer Selection for Deep Networks. *arXiv* **2023**, arXiv:2312.12781.
51. Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, K.-T.; Sun, J. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3296–3305.
52. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009. Available online: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf (accessed on 28 September 2026).
53. Haberer, J.; Wilhelm, J.E.; Landsiedel, O. Slimmable ConvNeXt: Width-Adaptive Inference for Efficient Multi-Device Deployment. *arXiv* **2026**, arXiv:2605.22677.

54. Shen, M.; Yin, H.; Molchanov, P.; Mao, L.; Liu, J.; Alvarez, J.M. Structural Pruning via Latency-Saliency Knapsack. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022; Volume 35. [CrossRef]
55. Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; Sun, J. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 544–560.
56. Mnih, V.; Heess, N.; Graves, A.; Kavukcuoglu, K. Recurrent Models of Visual Attention. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 8–13 December 2014; pp. 2204–2212.
57. Tang, Q.; Zhang, B.; Liu, J.; Liu, F.; Liu, Y. Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 777–786. [CrossRef]
58. Wang, Y.; Lv, K.; Huang, R.; Song, S.; Yang, L.; Huang, G. Glance and Focus: A Dynamic Approach to Reducing Spatial Redundancy in Image Classification. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020. [CrossRef]
59. Zhang, M.; Bu, T.; Hu, L. A Dynamic Dual-Processing Object Detection Framework Inspired by the Brain's Recognition Mechanism. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 6264–6274.
60. Yu, H.; Li, H.; Hua, G.; Huang, G.; Shi, H. Boosted Dynamic Neural Networks. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 10989–10997. [CrossRef]
61. Xie, L.; Yuille, A. Genetic CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1379–1388.
62. He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; Han, S. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 784–800.
63. Gao, Y.; Zhang, B.; Qi, X.; So, H.K.-H. DPACS: Hardware Accelerated Dynamic Neural Network Pruning through Algorithm-Architecture Co-Design. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Vancouver, BC, Canada, 25–29 March 2023; Volume 2, pp. 237–251. [CrossRef]
64. Xie, X.; Zhu, M.; Lu, S.; Wang, Z. Efficient Layer-Wise N:M Sparse CNN Accelerator with Flexible SPEC: Sparse Processing Element Clusters. *Micromachines* **2023**, *14*, 528. [CrossRef] [PubMed]
65. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 2820–2828.
66. Akhauri, Y.; Abdelfattah, M.S. On Latency Predictors for Neural Architecture Search. In Proceedings of the Seventh Annual Conference on Machine Learning and Systems (MLSys), Santa Clara, CA, USA, 13–16 May 2024.
67. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017*; ACM: New York, NY, USA, 2017; pp. 1–12. [CrossRef]
68. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
69. Papa, L.; Russo, P.; Amerini, I.; Zhou, L. A Survey on Efficient Vision Transformers: Algorithms, Techniques, and Performance Benchmarking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 7682–7700. [CrossRef] [PubMed]
70. Rao, Y.; Zhao, W.; Liu, B.; Lu, J.; Zhou, J.; Hsieh, C.-J. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–14 December 2021; Volume 34, pp. 13937–13949.
71. Lin, Z.; Wang, Y.; Zhang, J.; Chu, X. DynamicDet: A Unified Dynamic Architecture for Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 6282–6291.
72. Tang, S.; Wang, Y.; Kong, Z.; Zhang, T.; Li, Y.; Ding, C.; Wang, Y.; Liang, Y.; Xu, D. You Need Multiple Exiting: Dynamic Early Exiting for Accelerating Unified Vision Language Model. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 10781–10791.
73. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
74. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]

75. Bae, S.; Ko, J.; Song, H.; Yun, S.-Y. Fast and Robust Early-Exiting Framework for Autoregressive Language Models with Synchronized Parallel Decoding. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), Singapore, 6–10 December 2023; pp. 5910–5924. [CrossRef]
76. Ji, Y.; Wang, J.; Li, J.; Chen, Q.; Chen, W.; Zhang, M. Early Exit with Disentangled Representation and Equiangular Tight Frame. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 14128–14142. Available online: https://aclanthology.org/2023.findings-acl.889.pdf (accessed on 28 September 2026).
77. Wu, Z.; Xiong, C.; Ma, C.-Y.; Socher, R.; Davis, L.S. AdaFrame: Adaptive Frame Selection for Fast Video Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 1278–1287. [CrossRef]
78. Zhao, M.; Yu, Y.; Wang, X.; Yang, L.; Niu, D. Search-Map-Search: A Frame Selection Paradigm for Action Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023.
79. Wang, S.; Zhao, Q.; Wu, D.; Gao, J.; Tian, J. DyMAS-Net: Dynamic Multi-Scale Adaptive Sampling Network for Efficient Medical Image Segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2025*; Springer: Cham, Switzerland, 2025; pp. 96–105. [CrossRef]
80. Li, Z.; Lan, Y.; Lin, W. Footbridge Damage Detection Using Smartphone-Recorded Responses of Micromobility and Convolutional Neural Networks. *Autom. Constr.* **2024**, *166*, 105587. [CrossRef]
81. Lee, J.; Kim, H.; Yun, S.-J.; Choi, Y. Input-Adaptive Dynamic Neural Network for Efficient Object Detection Toward Resource-Constrained Deployment. *Electronics* **2026**, *15*, 2310. [CrossRef]
82. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
83. Haque, M.; Chauhan, A.; Liu, C.; Yang, W. ILFO: Adversarial Attack on Adaptive Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
84. Chen, S.; Chen, H.; Haque, M.; Liu, C.; Yang, W. The Dark Side of Dynamic Routing Neural Networks: Towards Efficiency Backdoor Injection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 24585–24594.
85. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A System for Large-Scale Machine Learning. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
86. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
87. Reddi, V.J.; Cheng, C.; Kanter, D.; Mattson, P.; Schmuelling, G.; Wu, C.-J.; Anderson, B.; Breughe, M.; Charlebois, M.; Chou, W.; et al. MLPerf Inference Benchmark. In Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 30 May–3 June 2020; pp. 446–459. [CrossRef]
88. Kang, W. QoS-Aware Inference Acceleration Using Adaptive Depth Neural Networks. *IEEE Access* **2024**, *12*, 49329–49340. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.