A Survey on Lightweight Vision Models and Model Optimization Techniques

www.surveyx.cn

Abstract

The rapid advancement of AI technologies necessitates the development of lightweight vision models optimized for deployment on resource-constrained edge devices. This survey examines the key strategies and methodologies employed to reduce computational and memory requirements of deep learning models, particularly for computer vision tasks. It explores model compression techniques, quantization methods, efficient architecture design, neural network pruning, and model optimization, emphasizing their significance in enhancing performance while maintaining accuracy. The survey highlights the challenges of deploying AI models on edge devices, such as limited computational resources and energy efficiency, and discusses the role of model optimization techniques in overcoming these hurdles. Through a comprehensive analysis of various approaches, including pruning, quantization, and neural architecture search, the survey underscores the importance of balancing accuracy and resource efficiency. It also presents case studies demonstrating the practical applications of lightweight models in real-world scenarios. The conclusion addresses ongoing challenges and future research directions, emphasizing the need for advanced compression techniques, robust quantization methods, and adaptive pruning strategies to further enhance the capabilities of AI models in resource-constrained environments. By integrating these strategies, the field can advance towards more efficient, robust, and versatile AI solutions, enabling effective deployment on edge devices and expanding the horizons of computer vision applications.

1 Introduction

1.1 The Need for Edge AI

The rapid advancement of AI technologies has created a pressing demand for models that can efficiently operate on edge devices, largely due to the computational and memory complexities associated with deep neural networks (DNNs). Edge computing, which processes data close to its source, offers significant advantages such as reduced latency and enhanced privacy, making it a viable solution for real-time applications [1]. The proliferation of edge devices, including IoT and mobile platforms, necessitates the deployment of sophisticated AI models capable of functioning effectively without heavy reliance on cloud resources.

Deploying large-scale DNNs on resource-constrained devices poses significant challenges due to their high memory and energy consumption, exacerbated by the millions or billions of parameters involved [2]. This is particularly problematic for applications like face recognition on mobile devices, where limited computational resources hinder the deployment of high-accuracy models. The inefficiency of deploying such models on constrained devices further emphasizes the need for energy-efficient models that can operate within available memory limits [3].

The challenges associated with cloud-based and mobile-based black box model deployments, particularly regarding latency and privacy, highlight the necessity for benchmarks in Edge AI that optimize

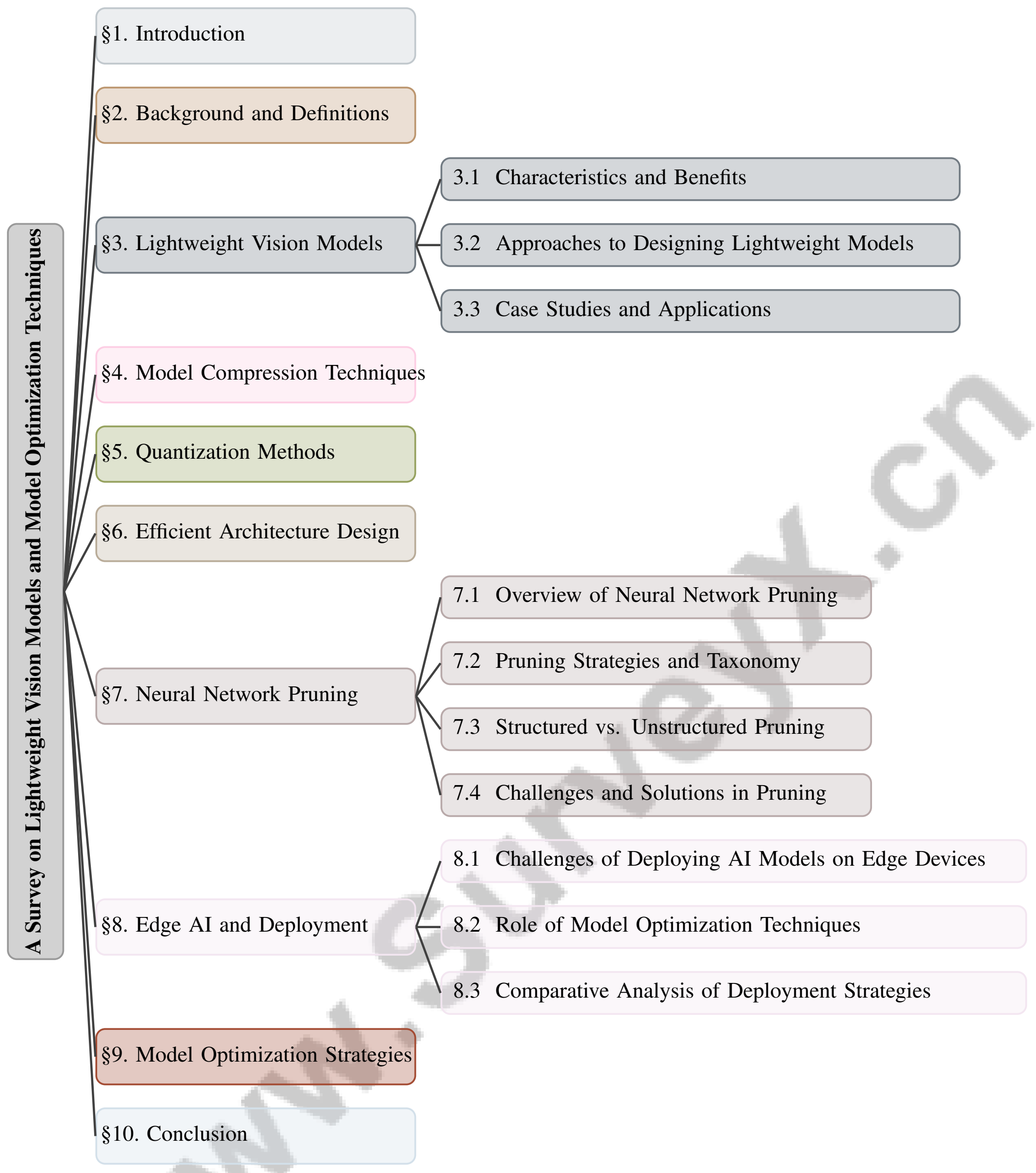

A Survey on Lightweight Vision Models and Model Optimization Techniques
§1. Introduction
§2. Background and Definitions
§3. Lightweight Vision Models
3.1 Characteristics and Benefits
3.2 Approaches to Designing Lightweight Models
3.3 Case Studies and Applications
§4. Model Compression Techniques
§5. Quantization Methods
§6. Efficient Architecture Design
§7. Neural Network Pruning
7.1 Overview of Neural Network Pruning
7.2 Pruning Strategies and Taxonomy
7.3 Structured vs. Unstructured Pruning
7.4 Challenges and Solutions in Pruning
§8. Edge AI and Deployment
8.1 Challenges of Deploying AI Models on Edge Devices
8.2 Role of Model Optimization Techniques
8.3 Comparative Analysis of Deployment Strategies
§9. Model Optimization Strategies
§10. Conclusion

Figure 1: chapter structure

computational resource utilization across mobile, edge, and cloud environments. Long inference times and high resource consumption of DNNs on embedded systems necessitate model compression techniques to make deep learning feasible for resource-constrained devices. Optimizing the inference process involves balancing on-device computational costs with communication overhead [1].

The excessive execution time of non-tensor layers in DNNs further impedes the real-time processing capabilities required for mobile applications. Consequently, there is a growing demand for AI models that can operate efficiently on edge devices while minimizing accuracy loss [2]. The benefits of deploying AI at the edge include reduced latency, enhanced privacy, and local data processing, which are critical for applications requiring real-time decision-making [3]. These advantages underscore the importance of developing AI models that efficiently address both computational and storage challenges in edge environments.

1.2 Structure of the Survey

This survey is organized into ten sections, each addressing distinct yet interconnected aspects of lightweight vision models and model optimization techniques. The introduction emphasizes the necessity of reducing computational and memory requirements in deep learning models, particularly for computer vision tasks, while highlighting their relevance in the context of edge AI. Section 2 provides background and definitions, clarifying key concepts such as lightweight vision models, model compression techniques, quantization methods, efficient architecture design, neural network pruning, edge AI, and model optimization.

Section 3 examines the characteristics and advantages of lightweight vision models, detailing various design strategies and methodologies. It presents successful implementations through case studies, demonstrating how lightweight Vision Transformers achieve high performance on minimal computational resources, addressing the challenges posed by traditional Convolutional Neural Networks in resource-constrained environments [4, 5, 6, 7]. Section 4 focuses on model compression techniques, including pruning, quantization, and low-rank approximation, providing a comparative analysis of their efficiency and effectiveness.

In Section 5, we analyze various quantization methods, emphasizing their role in lowering model precision while maintaining task accuracy. We explore distinct strategies, including variational network quantization, which frames the quantization process as a variational inference problem, allowing effective weight reduction without fine-tuning. Additionally, low-bit quantization techniques that convert full-precision networks to low-bitwidth integer versions are discussed, addressing challenges such as gradient mismatch and computational costs during training. This examination underscores quantization's significance in optimizing neural network performance for resource-constrained environments [8, 9]. Section 6 highlights efficient architecture design strategies, including neural architecture search (NAS) and innovative designs that optimize performance for constrained environments.

Section 7 focuses on neural network pruning, discussing various strategies and comparing structured and unstructured pruning methods while addressing associated challenges and solutions. Section 8 centers on deploying AI models on edge devices, emphasizing model optimization techniques such as quantization, partitioning, and early exit strategies. It includes case studies illustrating various edge AI applications, highlighting how these techniques balance latency and model performance in real-world scenarios. The section also addresses the complexities of implementing edge AI solutions, considering the constraints imposed by both edge computing and AI technologies, and outlines implications for MLOps engineers in selecting optimal deployment strategies [10, 11].

In Section 9, we analyze various model optimization strategies, including hyperparameter tuning, training optimizations, and inference acceleration. We examine adaptive optimization algorithms like AdamW, along with techniques such as mixed precision training and massively parallel computing, which enhance training efficiency and reduce memory footprint. We also explore model compression methods, including quantization, pruning, and knowledge distillation, highlighting their effectiveness in minimizing model size and inference delay while preserving accuracy. Inherent trade-offs between model accuracy and resource efficiency are discussed, particularly how larger models, despite their computational demands, can converge more quickly and are more resilient to compression techniques compared to smaller models. This section articulates the complexities and challenges in balancing these optimization strategies to enhance performance and efficiency in large-scale language models [12, 13, 14, 15]. The conclusion in Section 10 summarizes key points and reflects on future directions and potential advancements in the field.The following sections are organized as shown in Figure 1.

2 Background and Definitions

2.1 Key Concepts in Lightweight Vision Models

Lightweight vision models are pivotal for deploying AI on edge devices with limited computational resources, aiming to reduce computational and memory demands while maintaining performance [16, 15]. These models, crucial for mobile and IoT platforms, face challenges in balancing efficiency and accuracy, particularly in architectures like CNNs and ViTs [17].

Model compression techniques such as pruning and quantization are integral to lightweight model development. Pruning eliminates redundant parameters, aiding DNN deployment on constrained devices [18], while methods like Data-Driven Low-Rank (DDLR) reduce parameter counts in fully connected layers without sacrificing accuracy [19]. Quantization, which converts neural network weights and activations from floating-point to fixed-point, significantly reduces computational costs.

Developing compact models resilient to out-of-distribution (OOD) shifts requires advanced techniques to maintain accuracy despite size reductions [20]. Pruning methods must align with hardware architectures, such as crossbar-based accelerators, to ensure efficiency [21]. Efficient architecture design is vital to addressing the energy consumption and throughput demands of DNN computations [22]. Techniques like weight-sharing and edge inference enhance model efficiency by minimizing redundancy and optimizing edge device inference.

Knowledge distillation, where smaller student networks learn from larger teacher models, enhances accuracy retention while reducing model size [23]. This approach, along with training optimizations for high sparsity levels, continues to redefine capabilities in lightweight vision models [24]. Learning compressible representations in DNNs, as emphasized by Agustsson et al., is crucial for image and neural network compression applications, highlighting representation learning's role in lightweight model design [25].

Dynamic Data Pruning (DDP), introduced by Raju et al., optimizes training by selecting significant subsets of training samples [1]. These advancements in lightweight vision models integrate model compression techniques, efficient architecture design, and innovative quantization methods to achieve optimal performance within resource constraints, enhancing computer vision and enabling efficient AI solutions.

3 Lightweight Vision Models

The rising need for efficient AI solutions in resource-constrained settings has driven the advancement of lightweight vision models. These models are engineered to optimize performance and manage the deployment challenges of complex architectures. A thorough analysis of their features and benefits across various applications is crucial for understanding their influence. Figure 2 illustrates the hierarchical categorization of key concepts related to lightweight vision models, including their characteristics and benefits, approaches to design, and diverse applications. This figure highlights the efficiency and adaptability of these models in resource-constrained environments, showcasing strategies such as pruning and optimization, and their impact across various domains like mobile applications, healthcare, and smart surveillance. By integrating these insights, we can better appreciate the transformative potential of lightweight vision models in addressing contemporary challenges.

3.1 Characteristics and Benefits

Lightweight vision models are designed to minimize model size and enhance inference speed while preserving high accuracy, which is vital for low-resource settings [26]. Approaches like DeepRebirth improve speed and reduce energy consumption without sacrificing accuracy [27]. In edge AI applications, frameworks such as Deep-Edge highlight the advantages of lightweight models by adhering to latency constraints and demonstrating energy efficiency [28].

These models adapt to varying feature distributions, as demonstrated by the DFSE approach, which boosts classification accuracy across diverse datasets [29]. The learned global ranking (LeGR) methodology efficiently generates multiple ConvNet architectures with varying accuracy and speed, showcasing the flexibility of lightweight models [30]. Complexity-driven pruning methods compress CNNs, achieving competitive performance while significantly reducing training time and resource usage [31].

Techniques such as the Data-Driven Low-Rank (DDLR) method enable substantial parameter reduction and fast inference without specialized hardware, benefiting lightweight vision models in AI applications [19]. The ShuffleNet V2 architecture exemplifies superior speed and accuracy on mobile devices with a larger number of feature channels [32]. Additionally, CURL advances pruning in deep CNN models, often outperforming fine-tuned smaller models trained on larger datasets [33].

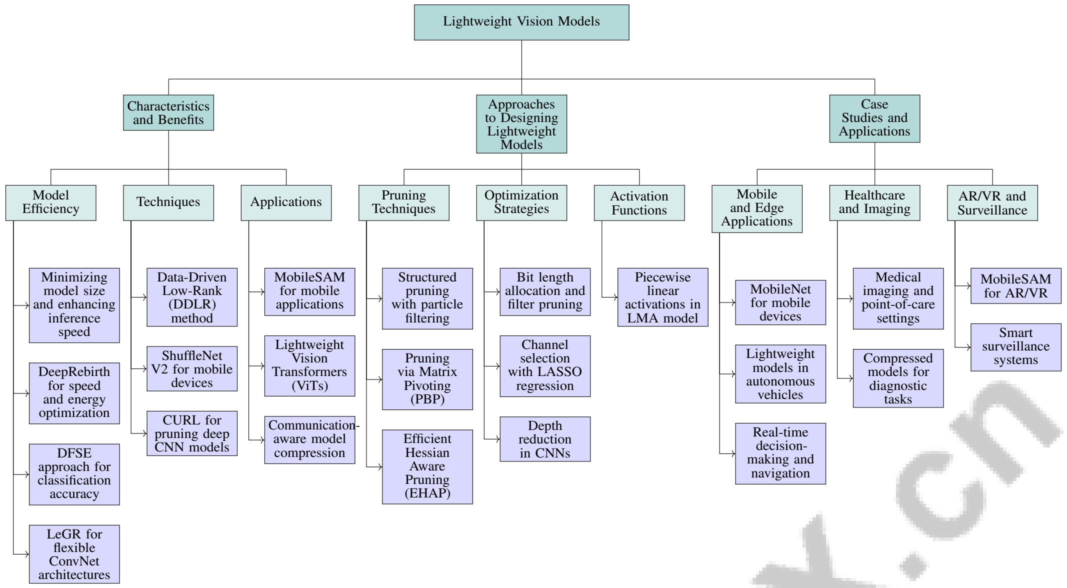

Lightweight Vision Models
Characteristics and Benefits
Approaches to Designing Lightweight Models
Case Studies and Applications
Model Efficiency
Techniques
Applications
Pruning Techniques
Optimization Strategies
Activation Functions
Mobile and Edge Applications
Healthcare and Imaging
AR/VR and Surveillance
Minimizing model size and enhancing inference speed
DeepRebirth for speed and energy optimization
DFSE approach for classification accuracy
LeGR for flexible ConvNet architectures
Data-Driven Low-Rank (DDLR) method
ShuffleNet V2 for mobile devices
CURL for pruning deep CNN models
MobileSAM for mobile applications
Lightweight Vision Transformers (ViTs)
Communication-aware model compression
Structured pruning with particle filtering
Pruning via Matrix Pivoting (PBP)
Efficient Hessian Aware Pruning (EHAP)
Bit length allocation and filter pruning
Channel selection with LASSO regression
Depth reduction in CNNs
Piecewise linear activations in LMA model
MobileNet for mobile devices
Lightweight models in autonomous vehicles
Real-time decision-making and navigation
Medical imaging and point-of-care settings
Compressed models for diagnostic tasks
MobileSAM for AR/VR
Smart surveillance systems

Figure 2: This figure illustrates the hierarchical categorization of key concepts related to lightweight vision models, including their characteristics and benefits, approaches to design, and diverse applications. It highlights the efficiency and adaptability of these models in resource-constrained environments, showcasing strategies such as pruning and optimization, and their impact across various domains like mobile applications, healthcare, and smart surveillance.

Lightweight models like MobileSAM offer practical benefits in mobile applications, achieving comparable performance to the original SAM while being over 60 times smaller and four times faster [34]. Similarly, lightweight Vision Transformers (ViTs) achieve high accuracy on small datasets without significant image scaling, surpassing existing transformer architectures with similar parameter counts [7]. Communication-aware model compression and task-oriented feature encoding further enhance efficiency by minimizing on-device computation and communication overhead [35].

Lightweight probabilistic networks quantify uncertainty in predictions and enhance robustness against adversarial examples [36]. ThiNet retains the original network structure, facilitating integration with existing frameworks while achieving substantial reductions in model size and computation [37]. Energy efficiency during inference tasks is a defining feature of lightweight models, particularly in edge environments with limited resources [38].

Moreover, the LMA model significantly enhances performance while maintaining low resource costs, making it suitable for computationally strict environments [39]. The framework proposed by Conejo et al. employs a coarse-to-fine optimization strategy, refining MAP estimation while reducing the label space based on classifier outputs [24].

As illustrated in Figure 3, the figure highlights the key categories and techniques in lightweight vision models, focusing on model compression, efficiency in edge AI, and advanced pruning methods. Each category showcases specific approaches that contribute to the development and deployment of efficient AI models in resource-constrained environments. These characteristics underscore the critical role of lightweight vision models in advancing AI applications, enabling efficient solutions in resource-constrained environments.

3.2 Approaches to Designing Lightweight Models

Designing lightweight vision models involves integrating strategies to enhance efficiency without sacrificing performance. Structured pruning, which removes network components to reduce complexity, is effective. Anwar et al. proposed a structured pruning method using particle filtering to assess network connections' importance, enabling effective pruning while preserving integrity [40]. Similarly, Sredojevic et al. introduced Pruning via Matrix Pivoting (PBP), enhancing computational efficiency while maintaining network representation flexibility [41].

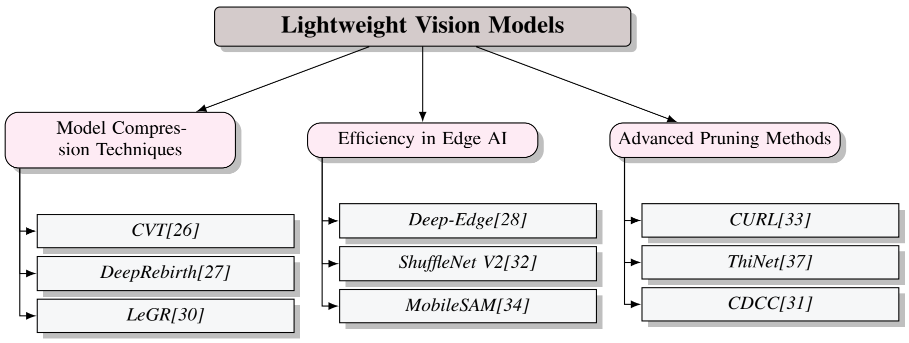


Figure 3: This figure illustrates the key categories and techniques in lightweight vision models, focusing on model compression, efficiency in edge AI, and advanced pruning methods. Each category highlights specific approaches that contribute to the development and deployment of efficient AI models in resource-constrained environments.

Optimizing bit length allocation and filter pruning as a neural architecture search (NAS) problem allows simultaneous optimization of these parameters for improved model efficiency [42]. He et al. developed an iterative two-step algorithm that selects channels using LASSO regression, followed by output reconstruction through linear least squares, effectively reducing model size while preserving performance [43].

Depth reduction in CNNs significantly enhances inference speed. Kim et al. noted that reducing CNN depth yields greater speed improvements than channel pruning, emphasizing depth optimization's importance in lightweight model design [44]. In mobile applications, Zhang et al. demonstrated the efficacy of a smaller image encoder in MobileSAM, optimizing for mobile environments while retaining the original mask decoder's functionality [34].

The use of piecewise linear activations, as in the LMA model, enhances expressiveness in neural networks without increasing model size. This approach suggests that increasing activation segments can improve performance while maintaining compactness [39]. Chong et al. introduced the Efficient Hessian Aware Pruning (EHAP) method, modifying existing Hessian-based pruning by utilizing FP16 precision for Hessian trace estimation, enabling faster computations and improved resource efficiency [45].

These methodologies highlight strategies for designing lightweight vision models. By emphasizing structured pruning, NAS optimization methods, depth reduction strategies, and innovative activation functions, researchers can create DNNs that maintain high accuracy while significantly reducing computational and storage demands. This approach is crucial for deploying large models, such as recent language and multimodal architectures, in resource-constrained environments, enhancing inference efficiency and addressing model over-parameterization challenges across various domains, including natural language processing and computer vision [46, 47, 48, 49].

3.3 Case Studies and Applications

Lightweight vision models, particularly Vision Transformers (ViTs), excel in real-world applications by achieving high performance on small datasets with minimal image scaling. These models can rival or surpass traditional CNNs in tasks like image classification and object detection under stringent resource constraints. Techniques such as pre-training with masked auto-encoders, alongside model compression methods like quantization and pruning, enhance efficiency, enabling deployment on resource-limited devices like UAVs and edge computing platforms. This advancement opens new possibilities in fields like surveillance and environmental monitoring, where accuracy and rapid inference are critical [4, 7, 50, 26, 6].

MobileNet exemplifies lightweight models in mobile applications, achieving significant computational cost and model size reductions while maintaining competitive accuracy. Its depthwise separable convolutions enable efficient processing on mobile devices, underscoring the practical benefits of lightweight models in such environments.

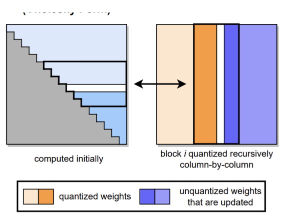
In autonomous vehicles, lightweight vision models enable real-time decision-making and navigation. Advanced neural network architectures allow rapid deployment of AI models capable of processing visual data, enhancing responsiveness to dynamic environments. Techniques like weight sharing and model pruning reduce computational resources for architecture search and inference, making sophisticated AI solutions feasible on resource-constrained edge devices. These systems effectively recognize and respond to real-time changes, improving operational efficiency and safety in smart surveillance [51, 52, 53, 54]. The YOLO framework has been adapted into a lightweight version for real-time applications in autonomous driving, demonstrating the feasibility of deploying complex vision tasks on resource-limited platforms.

Healthcare applications benefit from lightweight vision models, especially in medical imaging, where rapid analysis is essential. These models facilitate AI systems' deployment in point-of-care settings, enabling real-time diagnosis and decision support. Compressed models in diagnostic imaging tasks, such as tumor detection and classification, highlight lightweight architectures' efficacy in enhancing healthcare delivery, especially in resource-constrained environments. These models leverage advanced techniques like low-rank approximation and pruning to achieve compression rates of up to 95

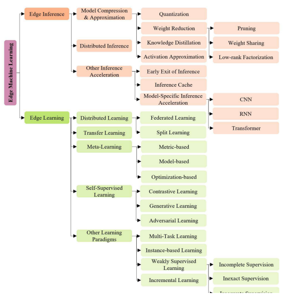
In AR and VR, lightweight models like MobileSAM enhance user experience by enabling seamless interactions and real-time processing. Optimized for resource-constrained devices, these models achieve performance levels comparable to heavier models while being over 60 times smaller and five times faster, allowing smooth operation on CPUs and facilitating advanced applications that require quick visual processing [34, 7]. Reducing latency is critical for maintaining immersive experiences in AR/VR applications, with techniques like model pruning and quantization ensuring manageable computational demands for consumer-grade hardware.

Lightweight vision models are effectively implemented in smart surveillance systems, enabling efficient video analysis and anomaly detection. Real-time video stream processing is essential for applications like security monitoring and traffic management. By utilizing advanced architectures and innovative compression techniques, such as Pruning-at-Initialization (PaI) and strategic training of larger models, these systems maintain continuous operation with significantly reduced computational requirements. This approach enables rapid generation of pruned DNNs that are up to 16.21 times smaller and twice as fast while preserving accuracy, enhancing efficiency in inference processes and delivering timely insights in edge computing environments [12, 55].

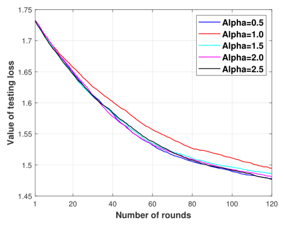
These case studies demonstrate diverse applications of lightweight vision models across domains, underscoring their significance in advancing AI capabilities in resource-constrained environments. By applying innovative design and optimization techniques, lightweight models enhance computer vision capabilities, enabling advanced deep learning technologies on devices like smartphones, robots, and IoT devices. These advancements are evident in the development of DCNNs and ViTs, optimized for performance while maintaining low computational costs. Consequently, lightweight models expand potential applications in areas such as smart home technology, telemedicine, and autonomous driving, achieving state-of-the-art performance on small datasets and challenging the traditional dominance of larger models [56, 5, 57, 7].


computed initially
block *i* quantized recursively column-by-column
quantized weights
unquantized weights that are updated

(a) Quantized Neural Network Training[58]



(b) Edge Machine Learning: A Comprehensive Overview[11]



(c) The graph shows the value of testing loss over the number of rounds for different values of alpha.[17]

Figure 4: Examples of Case Studies and Applications

AI-generated, for reference only.

As shown in Figure 4, the exploration of case studies and applications reveals significant insights into the practical implementations and advantages of lightweight vision models. The first case study, "Quantized Neural Network Training," delves into the intricate process of training neural networks using quantization techniques, illustrating how weights are computed and quantized in a structured manner to enhance computational efficiency. The second example, "Edge Machine Learning: A Comprehensive Overview," emphasizes various techniques, such as model compression and approximation, that facilitate efficient inference and learning at the edge. Lastly, the graph depicting testing loss over multiple rounds for varying alpha values underscores the nuances of model performance and optimization in lightweight vision applications. Together, these examples provide a comprehensive view of the strategies and methodologies employed in developing and deploying lightweight vision models, highlighting their potential to revolutionize machine learning and artificial intelligence [58, 11, 17].

4 Model Compression Techniques

4.1 Low-Rank Approximation and Factorization

Low-rank approximation and factorization are pivotal in compressing deep neural networks, focusing on reducing model complexity while preserving performance. These techniques decompose matrices and tensors, approximating key parameters to decrease model size and computational costs [59]. The Data-Driven Low-Rank (DDLR) method exemplifies this by applying low-rank structures to weight matrices in pretrained DNNs, achieving significant complexity reductions without compromising accuracy [19].

Integrating low-rank approximation with structured pruning enhances efficiency by removing entire channels, providing direct computational and storage benefits [60]. Automated pruning techniques like AutoCompress optimize pruning rates and weight reductions while preserving accuracy, addressing challenges such as narrowing the performance gap between compressed and uncompressed models and simplifying compression implementations [61, 23].

To illustrate the breadth of low-rank approximation techniques in model compression, Figure 5 presents a hierarchical classification that categorizes these methods into matrix decomposition, pruning strategies, and quantization and compression. This figure highlights the applications and innovations of each category, underscoring their significance in enhancing the efficiency of deep neural networks.

Combining low-rank approximation with joint optimization strategies allows simultaneous optimization of factorization and model learning, as seen in lossless joint low-rank factorization approaches [57]. Performance-aware pruning methods leverage low-rank techniques, achieving significant speedups compared to uninstructed channel pruning, which can lead to performance slowdowns [62].

Data-free methods, such as Data-Free Backbone Fine-tuning (DFBF), enhance low-rank approximation effectiveness by fine-tuning pruned backbones using synthetic images generated via backpropagation, eliminating reliance on original training data [63]. Similarly, Data-free Joint Rank-k Approximation compresses weight parameters by approximating matrices in linear layers without requiring additional calibration data [64].

Categorizing compression algorithms into high-cost and low-cost methods provides a framework for understanding the applicability and effectiveness of low-rank approximation techniques [65]. Integrating these methods with quantization strategies, such as SHVQ, minimizes combined distortion and entropy, enabling efficient compression without compromising model performance [25].

Advancements in low-rank approximation and factorization highlight their critical role in enhancing model compression techniques. By focusing on reducing computational demand while preserving essential model features, these methods facilitate efficient AI solutions in resource-constrained environments. Continuous optimization during training is exemplified by the gradual pruning of filters in the ASFP method [66], while Selective Weight Decay (SWD) employs a continuous pruning strategy that penalizes weights based on their significance, enhancing the pruning process [67].

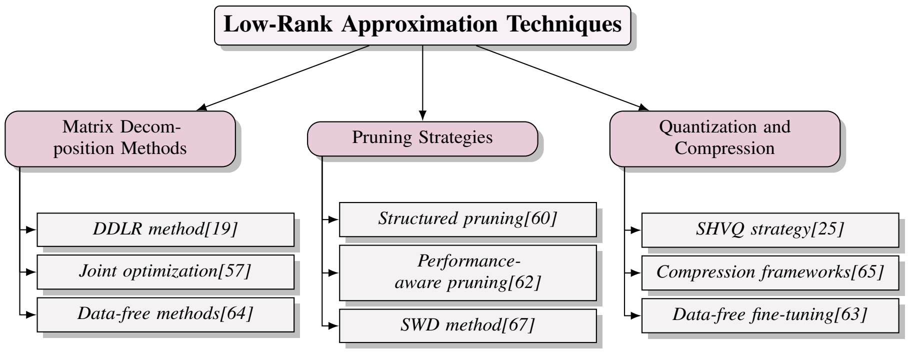

Low-Rank Approximation Techniques
Matrix Decomposition Methods
Pruning Strategies
Quantization and Compression
DDLR method[19]
Joint optimization[57]
Data-free methods[64]
Structured pruning[60]
Performance-aware pruning[62]
SWD method[67]
SHVQ strategy[25]
Compression frameworks[65]
Data-free fine-tuning[63]

Figure 5: This figure illustrates the hierarchical classification of low-rank approximation techniques used in model compression. It categorizes methods into matrix decomposition, pruning strategies, and quantization and compression, highlighting their applications and innovations in deep neural network compression.

Benchmark	Size	Domain	Task Format	Metric
PQ-Bench[68]	1,000,000	Deep Learning	Model Compression	SNR, MSE
RCNN-OD[69]	5,000	Object Detection	Object Localization And Classification	mAP, OOD Accuracy
RCP[70]	1,000,000	Image Classification	Channel Pruning	Top-1 Error, Top-5 Error
FP8[71]	75	Quantization	Model Evaluation	Pass Rate, Accuracy
EEDLQ[72]	1,000	Natural Language Processing	Text Generation	t/mWh, PPL
PIE[73]	1,000,000	Image Classification	Classification	Top-1 Accuracy, Top-5 Accuracy
ViT-CMP[4]	60,000	Image Classification	Image Classification	Model Size, Inference Speed
CP[74]	11,530	Object Detection	Detection	mAP, Latency

Table 1: This table provides a comprehensive overview of various benchmarks used in evaluating model compression techniques across different domains. It details the size, domain, task format, and metrics associated with each benchmark, offering insights into their applicability and effectiveness in model compression research.

4.2 Comparative Analysis of Compression Techniques

Model compression techniques such as pruning, quantization, and low-rank approximation are vital for alleviating the computational demands of deep learning models, especially in resource-constrained environments. The effectiveness of these techniques varies significantly based on application and hardware constraints. For instance, pruning methods like Asymptotic Structured Filter Pruning (ASFP) demonstrate superior performance by achieving substantial reductions in FLOPs with minimal accuracy degradation, underscoring the potential of structured pruning for efficient model compression [66].

As illustrated in Figure 6, which depicts the hierarchical categorization of model compression techniques, key methods in pruning, quantization, and low-rank approximation are highlighted, providing a visual framework for understanding these approaches. Additionally, Table 1 presents a detailed comparison of representative benchmarks employed in the evaluation of model compression techniques, highlighting their size, domain, task format, and associated metrics. Quantization techniques effectively reduce model precision and size but often encounter challenges in uniform application across layers. Differentiable fine-grained quantization addresses this by accounting for layer-specific sensitivity, thereby minimizing accuracy loss and enhancing overall model performance [75]. The integration of mixed-precision strategies further improves efficiency by balancing computational costs and precision, leading to better accuracy retention compared to traditional quantization methods [76].

Low-rank approximation complements these techniques by simplifying models through matrix decomposition, retaining essential information while reducing complexity. The Data-free Joint Rank-k Approximation method exemplifies this by achieving efficient compression without additional calibration data, as demonstrated with models like LLaMA-7B [64]. This joint optimization process effectively mitigates factorization errors while maintaining model performance, showcasing the potential of low-rank methods in model compression.

Comparative analyses of these techniques often evaluate trade-offs between compression rates and accuracy. Performance assessments using metrics such as PSNR and MS-SSIM against baseline methods like JPEG and JPEG 2000 reveal the effectiveness of compression strategies in preserving reconstruction quality [25]. However, challenges persist in maintaining performance across various sparsity levels without incurring additional computational costs or necessitating multiple training runs [77].

In diverse hardware contexts, the choice of compression technique significantly impacts performance, emphasizing the importance of selecting strategies that align with hardware capabilities and application requirements. For example, SWD employs a selective weight decay that adapts during training, contrasting with traditional methods that apply a uniform approach across the network [67]. Understanding these trade-offs enables researchers and practitioners to optimize AI models for efficient deployment across applications, ensuring effective performance in resource-constrained environments.

The vulnerability of compressed models to adversarial attacks is a critical consideration. Experiments indicate that pruned models exhibit heightened sensitivity to attacks, while quantized models are more vulnerable than their original counterparts, necessitating robust compression techniques capable of withstanding adversarial challenges [50]. The comparative analysis of compression techniques underscores the necessity for innovative strategies that balance efficiency, accuracy, and robustness in AI model deployment.

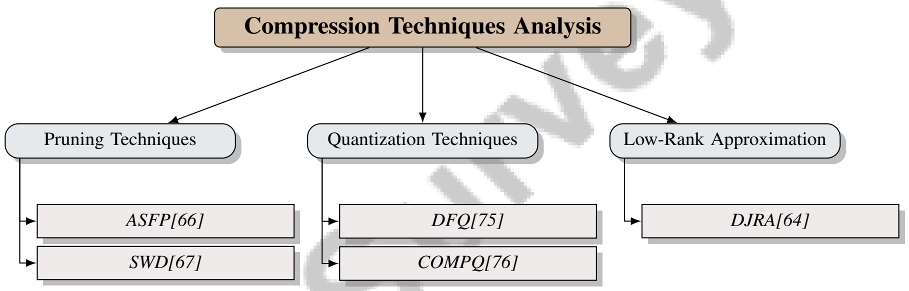


Figure 6: This figure illustrates the hierarchical categorization of model compression techniques, highlighting key methods in pruning, quantization, and low-rank approximation.

5 Quantization Methods

Deploying deep learning models efficiently on resource-limited devices is challenging. Quantization methods address this by reducing model size and computational demands while maintaining performance. The following subsection, "Introduction to Quantization," will delve into quantization's foundational principles, highlighting its crucial role in optimizing deep neural networks (DNNs) for environments with limited hardware resources. This sets the stage for exploring quantization techniques' practical applications and implications.

5.1 Introduction to Quantization

Quantization transforms high-precision floating-point numbers into lower-precision formats like integers, reducing model size and computational requirements. This process is essential for deploying DNNs on devices with limited resources, where high memory usage and computational demands are significant hurdles [16]. By mapping continuous real-valued numbers into a discrete set, quantization minimizes memory footprint and inference time, thus facilitating efficient AI deployment in constrained environments [3].

The Fully Quantized Network (FQN) method exemplifies quantization's effectiveness, enabling object detection networks to operate with only 4-bit integer arithmetic, significantly cutting down model size and computational needs [78]. Similarly, Differentiable Fine-grained Quantization (DFQ) optimizes bitwidths per layer using gradient descent, allowing a mixed-precision scheme that balances computational cost with precision [75]. These methods demonstrate quantization's adaptability across various model architectures, enhancing efficiency without sacrificing accuracy.

Quantization also tackles the dynamic range issues in neural network weights. While low-bit quantization of pre-trained models reduces size, it may introduce errors, especially in mid-layers [79]. Techniques like Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) address these challenges, preserving accuracy while enabling effective quantization [3].

Evaluating quantization methods using benchmarks, such as Mean Squared Quantization Error (MSQE)-based and gradient-based methods, is crucial for optimizing weight quantization in hardware-friendly applications [80]. These evaluations ensure compatibility with hardware capabilities, aiding efficient deployment on devices with limited resources.

Beyond efficiency, quantization affects neural networks' adversarial robustness. Understanding how quantization influences resilience against adversarially-perturbed images is vital for developing robust AI systems [81]. As quantization techniques evolve, their role in advancing efficient AI deployment, especially in edge computing environments, remains critical. Tailoring quantization strategies to specific applications and hardware constraints ensures models perform efficiently while aligning with deployment hardware capabilities.

5.2 Quantization in Practice

Quantization is crucial for optimizing DNNs in resource-constrained environments, reducing model size and computational demands while maintaining performance. Techniques like weight, activation, and gradient quantization make DNNs suitable for devices with limited resources [82]. The effectiveness of quantization methods is often assessed using datasets like ImageNet, serving as a benchmark for models such as MobileNetV1 and MobileNetV2 [80].

Quantization Networks utilize a linear combination of Sigmoid functions with learnable parameters to efficiently quantize weights and activations, reducing complexity while maintaining performance [8]. This approach has been successful on ImageNet with models like ResNet-18, highlighting quantization's adaptability across architectures [83].

The Pruning for Quantization (PfQ) method improves fine-tuning of quantized DNNs by pruning weights that do not contribute positively to learning, enhancing performance without added complexity [79]. Zhao et al.'s approach combines shift and recentralized quantization to effectively quantize sparse CNN weights, demonstrating quantization's versatility [84].

Adaptive quantization methods manage neural networks' dynamic weight range effectively. Zhou et al. achieved improved performance through adaptive methods compared to equal bit-width quantization in experiments on DNN models pre-trained on ImageNet [85]. Yang et al.'s SLB method eliminates estimated gradients by using a differentiable approach to search for low-bit weights, enhancing optimization accuracy [86].

In hardware-friendly applications, evaluating quantization methods through benchmarks like MSQE-based methods ensures alignment with hardware capabilities, facilitating efficient deployment on limited-resource devices [80]. The Fully Quantized Network (FQN) achieves state-of-the-art performance in object detection with fully quantized 4-bit models, achieving minimal accuracy loss without specialized hardware [78], underscoring quantization's potential for efficient AI deployment.

Quantization-aware training strategies, such as the Constraint-Guided Model Quantization (CGMQ) algorithm, optimize bit-widths while ensuring compliance with computational constraints [87]. This approach is vital for aligning quantization with hardware capabilities, facilitating efficient deployment on limited-resource devices. The low-bit precision linear quantization framework introduced by Choukroun et al. demonstrates state-of-the-art results in neural network inference with minimal accuracy degradation, enabling efficient deployment in constrained environments [88].

Advanced quantization techniques, including FP8 formats and adaptive quantization, optimize computational efficiency and performance. Recent studies show these methods reduce memory footprint and latency for large models in natural language processing and computer vision while maintaining near-floating point accuracy. FP8 formats enhance workload coverage and model accuracy compared to traditional INT8 quantization, benefiting deployment in resource-constrained environments like edge devices. Frameworks like the AI Model Efficiency Toolkit (AIMET) streamline quantization, ensuring high performance across diverse applications [89, 85, 71, 58, 90]. By tailoring quantization strategies to specific applications and hardware constraints, models can be optimized for efficient performance, aligning with deployment hardware capabilities.

6 Efficient Architecture Design

6.1 Frameworks for Efficient Architecture Design

Designing efficient neural network architectures is essential for optimizing performance in resource-constrained environments. Recent advancements in automated design frameworks have significantly reduced the computational demands of traditional methods. One-shot architecture search, for instance, employs weight sharing to minimize training costs across multiple architectures. Techniques like Structurally Prune Anything (SPA) enhance model efficiency through structured pruning, reducing the need for extensive retraining [60, 52]. Neural Architecture Search (NAS) further automates this process by exploring a wide configuration space to find optimal structures that balance performance and efficiency, facilitating deployment on constrained devices.

Evolutionary algorithms in NAS, as shown by Real et al., iteratively refine architectures based on performance metrics, offering a more resource-efficient exploration compared to traditional methods [91, 92, 93, 52, 47]. Platforms like Auto-Keras simplify NAS implementation, allowing users to explore architectures with minimal manual input.

Frameworks such as the TensorFlow Model Optimization Toolkit (TF-MOT) and PyTorch's TorchVision library provide tools for model optimization through pruning and quantization, supporting efficient deployment on edge devices [12, 94]. Hardware-aware NAS frameworks, like FBNet, incorporate hardware constraints into the search process, optimizing model design for edge computing scenarios [35, 11, 95, 55]. Lightweight building blocks, such as depthwise separable and group convolutions, further reduce complexity while maintaining performance, crucial for mobile architectures like MobileNet and ShuffleNet [32]. Collectively, these frameworks enhance neural network performance, significantly lowering the computational burden of traditional architecture search methods and facilitating the development of high-performance models that meet deployment platform constraints.

6.2 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) is a forefront methodology for automatically designing neural network architectures, optimizing for performance within resource constraints. NAS explores extensive search spaces to identify architectures that balance accuracy, latency, and efficiency. Differentiable Neural Architecture Search for Compression (DNAS-C) exemplifies this by optimizing bit allocation and filter pruning using a differentiable search process [42]. The One-Shot Architecture Search (OSAS) method further reduces computational overhead by evaluating architectures through weight sharing [52].

NAS-based pruning methods incorporate diverse strategies like weight-dependent criteria and activation-based methods, tailoring architectures for specific scenarios [96]. By integrating NAS with structured pruning, redundant components are systematically removed, optimizing performance. NAS also addresses latency by replacing inefficient layers with identity functions and merging operations, reducing latency in resource-constrained environments [44].

NAS offers significant benefits in designing optimized architectures by employing differentiable searches, weight sharing, and structured pruning. This approach addresses the challenges of modern neural networks' size and demands, reducing memory footprint and complexity. Furthermore, NAS optimizes all stages of model development, enhancing sparsification and facilitating deployment in constrained environments [48, 97, 2, 47, 49]. As NAS research progresses, its role in shaping efficient neural network design remains pivotal.

6.3 Innovative Architectural Designs

Innovative architectural designs enhance neural network efficiency and performance, particularly in resource-constrained environments. These designs streamline architectures, reduce complexity, and maintain accuracy. The One-Shot Architecture Search (OSAS) method demonstrates effective architecture search without hypernetworks or reinforcement learning, using a one-shot model to evaluate architectures through weight sharing, thus reducing computational overhead [52]. This approach efficiently explores architectural configurations, identifying designs that balance performance and resource efficiency [60, 52, 14, 34].

Lightweight building blocks, such as depthwise separable and group convolutions, further enhance efficiency by reducing DNN computational demands while preserving expressiveness. This balance is crucial for mobile-friendly architectures like MobileNet and ShuffleNet, which employ compression and pruning for optimal performance on constrained devices [32, 98, 99, 55]. These architectures ensure high accuracy and rapid inference, suitable for edge AI applications.

Figure 7 illustrates the categorization of innovative architectural designs in neural networks, highlighting efficient architecture search methods, mobile-friendly architectures, and hardware-aware strategies. Each category emphasizes specific techniques and frameworks that enhance performance and efficiency in resource-constrained environments. Hardware-aware strategies, like those in the FBNet framework, align innovations with hardware capabilities, enhancing performance and efficiency across applications. This includes model compression, optimizing hardware accelerators, and addressing security through mechanisms like homomorphic encryption. Selecting deployment operators and tiers is crucial for balancing latency and accuracy in constrained environments. These strategies facilitate efficient and secure DNN implementation in diverse contexts [10, 95, 100].

Innovative architectural designs in neural networks advance efficiency and performance by streamlining search processes, reducing costs through weight sharing, and enhancing neural representation understanding. These developments enable high-performing architectures without extensive resources and inform new pruning methods that optimize resource use. Consequently, the field is evolving significantly, particularly in resource-constrained environments like Edge AI, where balancing compute demands and memory bandwidth is critical [101, 52, 47, 22]. Through one-shot models, lightweight components, and hardware-aware strategies, these designs create powerful, resource-efficient models, advancing AI capabilities in constrained environments. As research progresses, these approaches will remain pivotal in shaping efficient AI solutions.

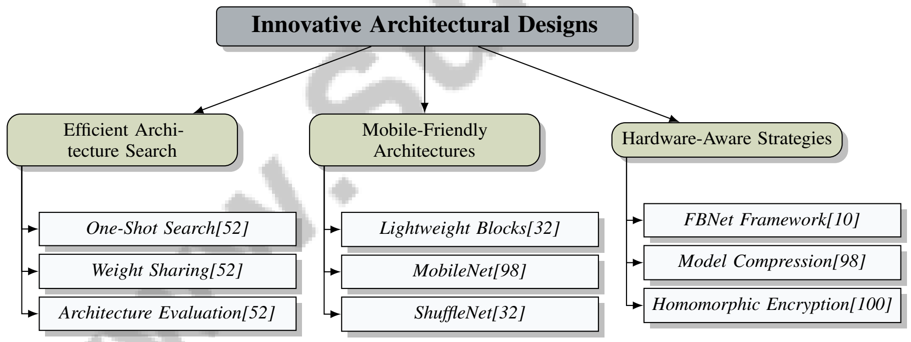

Innovative Architectural Designs
Efficient Architecture Search
Mobile-Friendly Architectures
Hardware-Aware Strategies
One-Shot Search[52]
Weight Sharing[52]
Architecture Evaluation[52]
Lightweight Blocks[32]
MobileNet[98]
ShuffleNet[32]
FBNet Framework[10]
Model Compression[98]
Homomorphic Encryption[100]

Figure 7: This figure illustrates the categorization of innovative architectural designs in neural networks, highlighting efficient architecture search methods, mobile-friendly architectures, and hardware-aware strategies. Each category emphasizes specific techniques and frameworks that enhance performance and efficiency in resource-constrained environments.

7 Neural Network Pruning

7.1 Overview of Neural Network Pruning

Neural network pruning is a pivotal optimization strategy in deep learning, targeting the reduction of model size and computational demands while maintaining or enhancing performance. This is crucial for deploying models on resource-constrained platforms like mobile and edge devices, where large models pose significant challenges [102]. By selectively removing unnecessary weights, neurons, or layers, pruning streamlines models for efficient deployment.

Layer-adaptive channel pruning exemplifies a method that identifies and prunes redundant filters, focusing on areas that significantly reduce redundancy without losing essential features [102]. The

Structurally Prune Anything (SPA) method further enhances pruning by aggregating interdependent channels, transforming various criteria into structured algorithms applicable at any training stage [60].

Pruning techniques also improve network robustness. For instance, Li et al. proposed a method that enhances certified robustness without compromising standard accuracy, achieving better verified accuracies across datasets [103]. This dual benefit highlights pruning's role in enhancing both efficiency and robustness.

In federated learning, frameworks like FedTiny create optimized tiny models for devices with limited memory and computational capabilities [104]. The Stochastic Frank-Wolfe algorithm further optimizes neural networks under norm constraints, enhancing compression stability and pruning outcomes [77].

Pruning typically involves training a model to convergence, applying magnitude-based pruning to create a sparse mask, and retraining the sparse model to optimize performance, as shown in end-to-end network pruning pipelines [2]. This iterative approach ensures pruned models maintain performance while benefiting from reduced complexity.

Challenges persist, particularly with sub-optimal performance when pruning in a single dimension, which can lead to excessive pruning and accuracy loss [105]. Addressing these requires comprehensive strategies that consider multiple dimensions and criteria.

Neural network pruning remains a critical strategy for optimizing model efficiency and performance. By leveraging diverse techniques and frameworks, pruning facilitates advanced AI solutions' deployment in resource-limited environments, ensuring models are both effective and efficient. Automated pruning techniques, such as the AE-BERT framework, promise to enhance AI model performance, particularly in resource-constrained settings, by optimizing model size and inference speed while maintaining high accuracy across applications like NLP and computer vision [48, 106, 46, 49].

7.2 Pruning Strategies and Taxonomy

Pruning strategies are vital for enhancing neural network efficiency by selectively reducing parameters while preserving performance. These strategies include weight pruning, neuron pruning, and channel pruning [93]. Each presents unique advantages and trade-offs concerning computational efficiency and model accuracy.

Weight pruning, a common technique, removes individual weights based on magnitude, achieving sparsity without altering the network's overall structure. Vysogorets et al. introduced a benchmark emphasizing effective sparsity, accounting for inactive connections for realistic model compression assessment [107]. However, weight pruning can result in a fragmented structure that may not fully leverage hardware acceleration.

Neuron pruning focuses on removing entire neurons or filters, leading to substantial network size reductions. The Incredible Shrinking Neural Network method exemplifies this approach, achieving significant size reduction without considerable performance loss [47]. This method effectively reduces computational overhead, improving inference speed, making it suitable for resource-constrained deployments.

Channel pruning, as discussed by Wang et al., involves removing entire channels within a layer, significantly enhancing computational efficiency [102]. This strategy balances pruning speed and model accuracy, though careful channel selection is necessary to avoid performance degradation. The SPA method refines this by constructing a computational graph, identifying coupled channels, and applying structured pruning based on importance scores [60].

Dynamic pruning strategies, such as Dynamic Probabilistic Pruning (DPP), integrate pruning and quantization within a unified framework, allowing flexible pruning at various granularity levels [108]. This adaptability facilitates strategies that adjust to changing computational constraints and application requirements. The iterative Combinatorial Brain Surgeon (iCBS) method enhances traditional pruning by solving optimization problems iteratively over subsets of weights, improving scalability and effectiveness [109].

The complexity of identifying optimal pruning strategies is underscored by the NP-Hard nature of pruning in over-parameterized networks, as highlighted by Qian et al. [110]. Advancements in matrix sparsification algorithms, such as those proposed by Yao et al., enhance interpretability and performance of pruned networks by preserving the spectral properties of the original model [111].

The taxonomy of pruning strategies reflects the diversity of approaches for optimizing neural networks. By categorizing strategies according to their distinct characteristics and applications, researchers can make informed decisions about selecting suitable methods for unique deployment scenarios. This systematic approach enhances model effectiveness and efficiency in resource-constrained environments, addressing critical factors such as latency, accuracy, and resource optimization essential for deploying large language models and other machine learning applications at the edge. Memory-efficient techniques and model compression significantly reduce operational costs and environmental impact, facilitating sustainable AI advancements [11, 10, 17].

7.3 Structured vs. Unstructured Pruning

Structured and unstructured pruning are key methodologies in neural network optimization, each offering distinct advantages and challenges. Structured pruning systematically removes entire components, such as channels, filters, or layers, from the architecture. This method is beneficial for producing models amenable to hardware acceleration, maintaining a regular structure that aligns with existing computational frameworks [112]. Channel-wise pruning using a Lagrangian multipliers framework, for example, allows for gradual tapering of resources while fine-tuning the network, resulting in effective pruning without significant performance loss.

Unstructured pruning focuses on eliminating individual weights based on criteria like magnitude, often achieving higher sparsity levels and greater reductions in model size and computational demands. However, the resulting irregular sparsity pattern can complicate efficient hardware implementation, as it does not align with typical data structures used in many processing units [113]. The Global MP method exemplifies a straightforward, magnitude-based approach that avoids complex heuristics, offering a simpler alternative to state-of-the-art methods requiring extensive hyper-parameter tuning.

The Dynamic Probabilistic Pruning (DPP) algorithm illustrates a hybrid approach, dynamically generating structured sparsity patterns based on probabilistic subsampling, enabling end-to-end optimization [108]. This adaptability allows leveraging benefits from both structured and unstructured pruning, optimizing the model for specific deployment scenarios.

Another innovative approach is the Feather module, which enhances the pruning process during standard training by employing a novel thresholding function and gradient scaling technique [114]. This method integrates pruning into the training process, facilitating real-time network optimization and reduced training time [115].

The choice between structured and unstructured pruning depends on the application and hardware constraints. Structured pruning is preferred for scenarios requiring efficient hardware implementation, while unstructured pruning suits applications where maximum sparsity is desired, and hardware constraints are less stringent. The Lottery Ticket Hypothesis further informs decision-making by suggesting smaller subnetworks, or "winning tickets," can be identified within a larger network, performing comparably to the full network when trained in isolation [110].

Both structured and unstructured pruning effectively optimize neural networks, presenting distinct advantages and challenges; structured pruning enhances hardware efficiency by maintaining model integrity while reducing computational costs, whereas unstructured pruning allows flexible weight removal but may complicate deployment on certain hardware platforms [96, 116]. Understanding these differences enables researchers and practitioners to tailor pruning strategies to meet specific application needs, ensuring models remain effective and efficient in diverse deployment environments.

7.4 Challenges and Solutions in Pruning

Neural network pruning is crucial for optimizing model efficiency, yet it faces challenges impacting its effectiveness in real-world applications. One primary challenge is the computational complexity in many pruning methods, such as Shapley Pruning, which requires calculating the Shapley value, hindering its application in large networks unless approximation techniques are employed [117]. Similarly, iterative optimization methods like iCBS face limitations due to substantial resource requirements, particularly for large models [109].

Extensive retraining after pruning is another significant challenge, often resource-intensive and time-consuming. This limitation is pronounced in methods requiring extensive hyper-parameter optimization to maintain model accuracy, especially for complex architectures. Enhancing scalable hyper-parameter optimization frameworks can address high computational costs associated with deep learning models. Efficient exploration of optimal configurations can reduce hyper-parameter tuning time by up to 37

The reliance on initial pruning strategy choices also presents a challenge, as suboptimal decisions can lead to less effective outcomes. While methods like EagleEye improve evaluation processes, they depend on the initial choice of pruning strategies, which may not always yield optimal results [118]. Adaptive techniques that adjust pruning strategies based on network feedback could enhance robustness and effectiveness.

In scenarios involving high sparsity levels, pruning methods may encounter training instability, affecting performance [119]. Addressing this requires developing robust sparsity control mechanisms that adapt to different network configurations, ensuring consistent performance.

Moreover, Prune-at-Inference (PaI) methods often lag behind Prune-at-Training (PaT) methods in performance, with many PaI approaches not yielding expected speedup or accuracy improvements [92]. Bridging this performance gap involves refining PaI techniques to align with efficiencies achieved by PaT methods.

Despite these challenges, pruning offers significant advantages, especially in resource-constrained environments. The FedTiny framework effectively mitigates bias in model pruning and enhances resource efficiency, resulting in higher accuracy and lower resource consumption [104]. Comprehensive strategies, like the method proposed by Wang et al., improve model accuracy and reduce computational costs, demonstrating potential for optimizing performance [105].

Addressing these challenges involves theoretical advancements, algorithmic innovations, and practical implementations. Advancing efficient algorithms, adaptive techniques, and scalable optimization frameworks can significantly improve neural networks' performance and resource efficiency. This progress is crucial given the growing demand for compressing large models for mobile deployment while maintaining accuracy. Recent findings indicate that simpler pruning methods, such as Global Magnitude Pruning, can outperform more complex state-of-the-art techniques. Additionally, standardized evaluation frameworks like ShrinkBench aim to address benchmarking inconsistencies, facilitating reliable comparisons of pruning methods and driving innovation in the field [113, 91, 97].

To further illustrate these concepts, Figure 8 categorizes the key challenges and solutions in neural network pruning into computational challenges, pruning strategies, and innovative frameworks. This figure highlights significant methods and frameworks within each category, providing insight into the current landscape of pruning techniques.

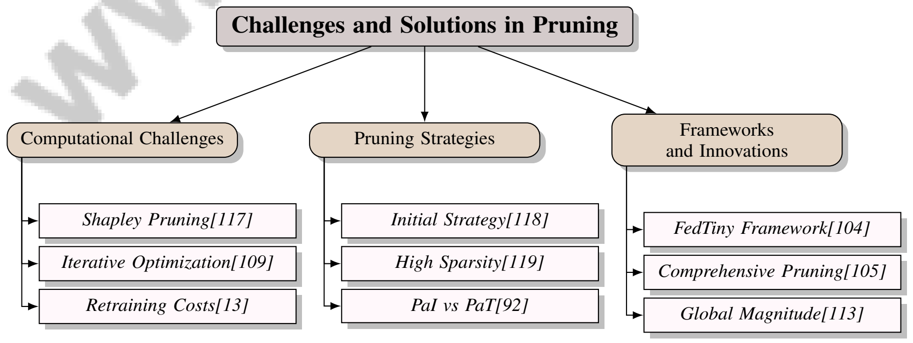


Figure 8: This figure illustrates the key challenges and solutions in neural network pruning, categorizing them into computational challenges, pruning strategies, and innovative frameworks. It highlights significant methods and frameworks in each category, providing insight into the current landscape of pruning techniques.

AI-generated, for reference only.

8 Edge AI and Deployment

8.1 Challenges of Deploying AI Models on Edge Devices

Deploying AI models on edge devices is challenged by limited computational resources, memory, and energy efficiency. Balancing model complexity with available resources is critical, as deep neural networks (DNNs) often necessitate substantial computational power and memory, which edge devices lack [2]. To address this, model compression techniques like pruning and quantization reduce model size and complexity while preserving performance [3].

Latency is crucial for real-time applications such as autonomous vehicles and video analytics, where low-latency inference is essential [1]. Edge computing mitigates latency by processing data closer to the source, reducing the need for data transmission to centralized servers [1].

Energy efficiency is vital for battery-operated edge devices. Techniques such as efficient architecture design and power-aware pruning lower energy consumption while maintaining accuracy [3]. Lightweight models like MobileNet and ShuffleNet exemplify energy-efficient AI solutions [32].

The heterogeneity of edge devices demands adaptable models for varying hardware specifications. Hardware-aware optimization and adaptive quantization enable models to dynamically adjust to resource constraints, enhancing flexibility across diverse environments [60].

Security and privacy concerns are paramount when processing data locally, necessitating data privacy and model security, especially with sensitive information. Techniques like federated learning and secure model updates enable decentralized training and updates without exposing sensitive data [104].

Addressing computational, latency, energy, and security challenges in AI model deployment on edge devices requires a multifaceted approach. Utilizing advanced model compression techniques, dynamic quantization, and efficient architecture design can create AI solutions that achieve high accuracy while significantly reducing model size and computational complexity. These methods can yield reductions of up to 89.7

8.2 Role of Model Optimization Techniques

Model optimization techniques are crucial for deploying AI models on edge devices, where computational resources, memory constraints, and energy efficiency are critical. These techniques encompass pruning, quantization, and efficient architecture design. As illustrated in Figure 9, the primary model optimization techniques for AI deployment on edge devices are categorized into these three main areas, with each category including key methods and frameworks that highlight their contributions to enhancing computational efficiency and model performance. Pruning reduces non-zero parameters in deep neural networks, significantly reducing model size with minimal accuracy loss, while quantization lowers the precision of weights and activations, often to 8-bit integers, decreasing computational requirements. Integrating these methods enhances efficiency and model accuracy [120, 68, 46].

Pruning techniques, such as the SPA framework, provide versatile solutions for reducing model complexity across various architectures, facilitating efficient AI model deployment [60]. Performance-aware channel pruning significantly optimizes convolutional neural networks (CNNs) for resource-constrained environments, ensuring high accuracy while minimizing computational demands [62].

Quantization methods, like Fully Quantized Networks (FQNs), address deployment challenges by reducing model precision and computational requirements [78]. These methods are particularly effective in environments with limited memory and processing power, enabling efficient deployment without substantial accuracy loss. The DDLR method showcases low-rank approximation techniques in compressing pretrained deep neural networks with minimal accuracy degradation, facilitating deployment on edge devices without retraining [19].

Efficient architecture design leverages structured weight representation and innovative strategies to reduce memory footprint and enhance inference performance. The LFPC method achieves over 60

In federated learning scenarios, model optimization techniques like those in the FedTiny framework demonstrate significant improvements in accuracy, computational efficiency, and memory usage, achieving a 2.61

Model optimization techniques are pivotal in advancing AI model capabilities for edge deployment. By integrating advanced techniques such as model compression, distributed inference, and structured pruning, AI solutions can be tailored to meet the rigorous demands of edge environments, optimizing resource utilization and minimizing latency for real-time applications across diverse use cases from computer vision to natural language processing [10, 106, 28, 11, 55]. As research progresses, these strategies will remain essential for enhancing AI model deployment in resource-constrained settings.

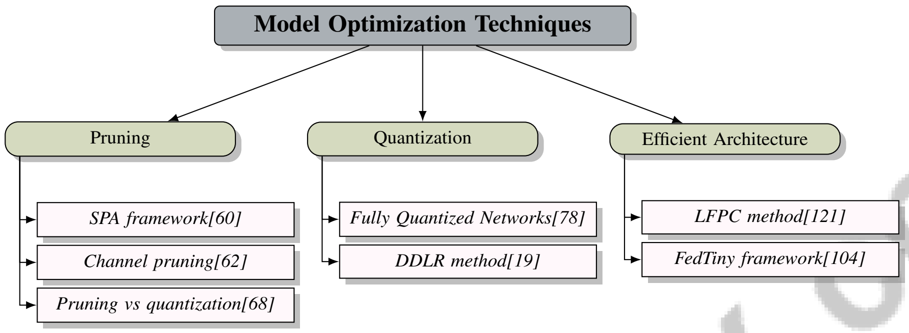


Figure 9: This figure illustrates the primary model optimization techniques for AI deployment on edge devices, categorized into pruning, quantization, and efficient architecture design. Each category includes key methods and frameworks, highlighting their contributions to enhancing computational efficiency and model performance.

8.3 Comparative Analysis of Deployment Strategies

Deploying AI models on edge devices necessitates diverse strategies, each offering unique advantages and challenges concerning computational efficiency, latency, and resource utilization. MLOps engineers must navigate critical decisions regarding deployment operators and tiers to meet specific latency and performance requirements. Hybrid strategies, such as combining quantization with early exit techniques, can significantly reduce latency while maintaining acceptable accuracy levels, particularly in resource-constrained settings. Conversely, when accuracy is paramount, solely employing quantization on edge devices is advisable, despite potential latency increases. The choice between mobile and edge deployment varies based on input data size and device computational capabilities, emphasizing the need for tailored approaches that consider communication and computation tradeoffs for optimal edge inference performance [10, 35]. A comparative analysis of these strategies is crucial for understanding their effectiveness in resource-constrained environments.

Model compression techniques, including pruning and quantization, are fundamental for reducing model size and computational demands. Pruning methods, such as structured and unstructured pruning, selectively remove network components to optimize model efficiency for edge deployment [60]. Quantization techniques, like those in Fully Quantized Networks, enhance deployment efficiency by converting model weights and activations to lower precision formats, significantly reducing memory usage and computational load [78].

Efficient architecture design, particularly through lightweight models like MobileNet and ShuffleNet, facilitates AI model deployment on edge devices. These architectures utilize depthwise separable convolutions and innovative design elements to minimize computational complexity while maintaining high performance [32]. Hardware-aware design frameworks, such as FBNet, ensure models are optimized for specific hardware platforms, maximizing performance and resource efficiency.

Deployment benefits from distributed learning frameworks like federated learning, which enable decentralized model training and updates, addressing privacy and security concerns by keeping data localized on edge devices [104]. Federated learning frameworks, exemplified by the FedTiny method, demonstrate substantial improvements in resource efficiency and model accuracy, underscoring their potential for edge deployment [104].

Edge computing architectures also play a crucial role by processing data closer to the source, reducing latency and bandwidth requirements. Leveraging edge computing allows AI models to achieve real-time inference capabilities, essential for applications such as autonomous vehicles and real-time video analytics [1].

9 Model Optimization Strategies

Optimizing neural network models involves a strategic combination of techniques to enhance their performance and efficiency, particularly on resource-constrained devices. A crucial aspect of this process is hyperparameter tuning alongside training optimizations, which shape the learning dynamics and overall effectiveness of neural networks in practical applications. The following subsections explore these components, emphasizing their role in developing robust AI models.

9.1 Hyperparameter Tuning and Training Optimizations

Hyperparameter tuning and training optimizations are essential for creating efficient neural network models, especially on devices with limited resources. Hyperparameter tuning involves optimizing parameters like learning rates and batch sizes, which significantly impact model performance [13]. The HPO-NP method exemplifies the use of pruned networks as proxies to expedite hyperparameter optimization, reducing computational demands [13]. Challenges arise in achieving precise parameter adjustments, as effectiveness varies with architecture and dataset [112]. Techniques like AMC enhance compression quality but require careful hyperparameter tuning for optimal performance across architectures [122]. Future research may focus on adaptive settings to dynamically adjust parameters based on training feedback [123].

Training optimizations are equally crucial, focusing on efficiency and effectiveness. The OBC framework applies OBS principles to weight pruning and quantization, reducing model size while maintaining accuracy [124]. The FLS method maintains performance at high sparsity by dynamically updating the inverse Fisher information matrix [125], while RSP provides a straightforward compression pipeline achieving high sparsity without extensive retraining [126]. Efficient pruning methods, such as those by Gurevin et al., ensure high accuracy during hard-pruning stages without extensive retraining [127]. AutoCompress streamlines training optimizations by automatically determining per-layer pruning rates to maximize weight reduction with minimal accuracy loss [61].

Integrating hyperparameter tuning and training optimizations—such as adaptive algorithms, mixed precision training, and pruning—is crucial for enhancing model performance and computational efficiency, especially in large-scale language models. These strategies accelerate convergence, reduce memory usage, and facilitate the deployment of compressed models that retain high accuracy, addressing the trade-offs between training and inference efficiency [12, 13, 15].

9.2 Inference Acceleration and Deployment on Edge Devices

Accelerating inference and deploying AI models effectively on edge devices are crucial to operating within computational, memory, and energy constraints. Pruned models significantly reduce computational load while maintaining performance, enhancing inference speed and conserving resources [13]. The RCT method exemplifies strategies for accelerating inference by optimizing models for edge devices, ensuring high accuracy with efficient deployment [16].

Quantization methods play a key role in accelerating inference on edge devices, converting high-precision numbers into lower-precision formats to reduce computational costs without sacrificing accuracy, highlighting their potential for deployment in resource-constrained environments [3].

The integration of pruning, resource-constrained training, and quantization provides a comprehensive framework for accelerating inference and deploying AI models on edge devices. These methods collectively address the challenges of limited computational resources and energy constraints, developing efficient AI solutions for real-world applications. As AI and edge computing research progresses, advanced strategies will be crucial for enhancing model deployment in edge environments, tackling challenges such as resource optimization, latency reduction, and privacy concerns with techniques like model compression, distributed inference, and efficient fine-tuning [10, 11, 17, 28].

9.3 Balancing Accuracy and Resource Efficiency

Balancing accuracy and resource efficiency in neural network models is critical, especially for deployment on resource-constrained devices. This involves maintaining high accuracy while minimizing computational and memory demands. Techniques like VNQ prepare neural networks for pruning and quantization, achieving significant model size reductions with minimal accuracy loss [9], eliminating the need for subsequent fine-tuning.

The Quant-Noise method maintains high accuracy levels at extreme compression rates, outperforming traditional methods like QAT by introducing noise during training to facilitate robust compression [128]. The SLB method effectively bridges the accuracy gap in quantized networks, enhancing performance [86].

Pruning techniques are essential for balancing accuracy and resource efficiency. Research by Iofinova et al. indicates high sparsity can be achieved without sacrificing accuracy, although bias amplification increases at high sparsity levels [56]. Structured pruning techniques, identified by He et al., offer promising research avenues, particularly in integrating pruning with other compression strategies [96].

Mixed-precision quantization methods adapt bit-width assignments based on layer sensitivity, improving performance while balancing accuracy and computational efficiency [76]. The FQN method stabilizes fine-tuning and optimizes quantization of weights and activations, mitigating instability and inaccuracy [78].

Findings from Zhu et al. show large-sparse models consistently outperform small-dense models, achieving significant parameter reductions with negligible accuracy loss [46]. This highlights pruning's effectiveness in maintaining performance while enhancing resource efficiency. The end-to-end network pruning pipeline by Dogariu et al. integrates training and pruning stages, differing from methods focusing on isolated techniques [2].

10 Conclusion

10.1 Challenges and Future Directions

Progress in lightweight vision models and optimization techniques is impeded by the need to maintain a balance between model compression and robustness. The integration of pruning and quantization presents a significant challenge, as improper application can degrade performance. Future research should aim to develop automated compression techniques that enhance both explainability and effectiveness. Additionally, a deeper understanding of scaling laws could provide valuable insights into optimizing model performance.

The impact of structured sparsity on execution time in convolutional networks is not well understood, yet it holds potential for improving computational efficiency without compromising accuracy. Evaluating pruned models across various applications could further demonstrate the adaptability and resilience of these strategies. In the realm of edge AI, optimizing performance models and achieving efficient load balancing among parameter servers are vital for effective deployment in resource-limited settings. Research should focus on refining library heuristics for diverse neural network architectures and exploring innovative pruning strategies to enhance edge AI applications. Integrating Resource-Constrained Training with various hardware architectures may address challenges related to bitwidth optimization.

The development of robust quantization techniques and quantization-aware training remains a priority. Future efforts should focus on refining these techniques, optimizing quantization-aware training methods, and addressing real-world deployment challenges. Exploring compression techniques on devices with limited resources is crucial for expanding the reach of lightweight models. Adaptive pruning methods that enhance performance without extensive tuning could significantly advance the field. Expanding datasets and refining pruning techniques through advanced methods will address specific application challenges more effectively.

The exploration of initialization methods, learned masks, and their application to larger models and complex environments presents a promising research avenue. Optimizing sampling methods in random pruning and investigating sophisticated channel configuration strategies could further enhance model performance. Additionally, further exploration of compression techniques and their impact on adversarial robustness, along with the development of new attack strategies, is warranted. Examining various neural network architectures, refining models to manage non-convex decision boundaries, and analyzing the role of activation functions in adversarial robustness are promising directions for future research.

The future of lightweight vision models and optimization techniques hinges on the integration of diverse methods, adaptive strategies, and expanded applications. Addressing these challenges and exploring these pathways will propel the field towards more efficient, robust, and versatile AI solutions.

References

[1] Ravi S Raju, Kyle Daruwalla, and Mikko Lipasti. Accelerating deep learning with dynamic data pruning, 2021.

[2] Evan Dogariu. An end-to-end network pruning pipeline with sparsity enforcement, 2023.

[3] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[4] Feiyang Chen, Ziqian Luo, Lisang Zhou, Xueting Pan, and Ying Jiang. Comprehensive survey of model compression and speed up for vision transformers, 2024.

[5] Hanhua Long, Wenbin Bi, and Jian Sun. Lightweight design and optimization methods for dcnns: Progress and futures. *arXiv preprint arXiv:2412.16886*, 2024.

[6] Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A survey on transformer compression, 2024.

[7] Jen Hong Tan. Pre-training of lightweight vision transformers on small datasets with minimally scaled images. *arXiv preprint arXiv:2402.03752*, 2024.

[8] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316, 2019.

[9] Jan Achterhold, Jan Mathias Koehler, Anke Schmeink, and Tim Genewein. Variational network quantization. In *International conference on learning representations*, 2018.

[10] Jaskirat Singh, Bram Adams, and Ahmed E. Hassan. On the impact of black-box deployment strategies for edge ai on latency and model performance, 2024.

[11] Wenbin Li, Hakim Hacid, Ebtesam Almazrouei, and Merouane Debbah. A comprehensive review and a taxonomy of edge machine learning: Requirements, paradigms, and techniques, 2023.

[12] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR, 2020.

[13] Kangil Lee and Junho Yim. Hyperparameter optimization with neural network pruning, 2022.

[14] Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. Search for efficient large language models, 2024.

[15] Taiyuan Mei, Yun Zi, Xiaohan Cheng, Zijun Gao, Qi Wang, and Haowei Yang. Efficiency optimization of large-scale language models based on deep learning in natural language processing tasks, 2024.

[16] Tian Huang, Tao Luo, Ming Yan, Joey Tianyi Zhou, and Rick Goh. Rct: Resource constrained training for edge ai, 2021.

[17] Yanjie Dong, Haijun Zhang, Chengming Li, Song Guo, Victor C. M. Leung, and Xiping Hu. Fine-tuning and deploying large language models over edges: Issues and approaches, 2024.

[18] Ao Ren, Tao Zhang, Yuhao Wang, Sheng Lin, Peiyan Dong, Yen kuang Chen, Yuan Xie, and Yanzhi Wang. Darb: A density-aware regular-block pruning for deep neural networks, 2019.

[19] Dimitris Papadimitriou and Swayambhoo Jain. Data-driven low-rank neural network compression, 2021.

[20] James Diffenderfer, Brian R. Bartoldson, Shreya Chaganti, Jize Zhang, and Bhavya Kailkhura. A winning hand: Compressing deep networks can improve out-of-distribution robustness, 2021.

AI-generated, for reference only.

[21] Ling Liang, Lei Deng, Yueling Zeng, Xing Hu, Yu Ji, Xin Ma, Guoqi Li, and Yuan Xie. Crossbar-aware neural network pruning, 2018.

[22] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[23] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models, 2024.

[24] B. Conejo, N. Komodakis, S. Leprince, and J. P. Avouac. Speeding-up graphical model optimization via a coarse-to-fine cascade of pruning classifiers, 2014.

[25] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. *Advances in neural information processing systems*, 30, 2017.

[26] Eric Youn, Sai Mitheran J, Sanjana Prabhu, and Siyuan Chen. Compressing vision transformers for low-resource visual learning, 2023.

[27] Dawei Li, Xiaolong Wang, and Deguang Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices, 2018.

[28] Anirban Bhattacharjee, Ajay Dev Chhokra, Hongyang Sun, Shashank Shekhar, Aniruddha Gokhale, Gabor Karsai, and Abhishek Dubey. Deep-edge: An efficient framework for deep learning model update on heterogeneous edge, 2020.

[29] Hsu-Hsun Chin, Ren-Song Tsay, and Hsin-I Wu. A high-performance adaptive quantization approach for edge cnn applications, 2021.

[30] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1518–1528, 2020.

[31] Muhammad Zawish, Steven Davy, and Lizy Abraham. Complexity-driven cnn compression for resource-constrained edge ai, 2022.

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

[33] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1458–1467, 2020.

[34] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

[35] Jiawei Shao and Jun Zhang. Communication-computation trade-off in resource-constrained edge inference. *IEEE Communications Magazine*, 58(12):20–26, 2021.

[36] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3369–3378, 2018.

[37] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[38] Seyyidahmed Lahmer, Aria Khoshsirat, Michele Rossi, and Andrea Zanella. Energy consumption of neural networks on nvidia edge boards: an empirical model, 2022.

[39] Zhenhui Xu, Guolin Ke, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Light multi-segment activation for model compression, 2019.

[40] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.

[41] Ranko Sredojevic, Shaoyi Cheng, Lazar Supic, Rawan Naous, and Vladimir Stojanovic. Structured deep neural network pruning via matrix pivoting, 2017.

[42] Yochai Zur, Chaim Baskin, Evgenii Zheltonozhskii, Brian Chmiel, Itay Evron, Alex M. Bronstein, and Avi Mendelson. Towards learning of filter-level heterogeneous compression of convolutional neural networks, 2019.

[43] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[44] Jinuk Kim, Yeonwoo Jeong, Deokjae Lee, and Hyun Oh Song. Efficient latency-aware cnn depth compression via two-stage dynamic programming, 2023.

[45] Jack Chong, Manas Gupta, and Lihui Chen. Resource efficient neural networks using hessian based pruning, 2023.

[46] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[47] Aditya Sharma, Nikolas Wolfe, and Bhiksha Raj. The incredible shrinking neural network: New perspectives on learning representations through the lens of pruning, 2017.

[48] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations, 2024.

[49] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[50] Swapnil Parekh, Devansh Shah, and Pratyush Shukla. Attacking compressed vision transformers, 2022.

[51] Zhenyu Wang and Shahriar Nirjon. Characterizing disparity between edge models and high-accuracy base models for vision tasks, 2024.

[52] d simplifying one-shot architect.

[53] August Lidfelt, Daniel Isaksson, Ludwig Hedlund, Simon Åberg, Markus Borg, and Erik Larsson. Enabling image recognition on constrained devices using neural network pruning and a cyclegan, 2020.

[54] Hasib-Al Rashid and Tinoosh Mohsenin. Tinym^{2}net-v3: Memory-aware compressed multi-modal deep neural networks for sustainable edge deployment, 2024.

[55] Bailey J. Eccles, Leon Wong, and Blesson Varghese. Rapid deployment of dnns for edge computing via structured pruning at initialization, 2024.

[56] Eugenia Iofinova, Alexandra Peste, and Dan Alistarh. Bias in pruned vision models: In-depth analysis and countermeasures, 2023.

[57] Boyang Zhang, Daning Cheng, Yunquan Zhang, Fangmin Liu, and Jiake Tian. Lossless model compression via joint low-rank factorization optimization, 2024.

[58] Jiedong Lang, Zhehao Guo, and Shuyu Huang. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE, 2024.

[59] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[60] Xun Wang, John Rachwan, Stephan Günnemann, and Bertrand Charpentier. Structurally prune anything: Any architecture, any framework, any time, 2024.

[61] Ning Liu, Xiaolong Ma, Zhiyuan Xu, Yanzhi Wang, Jian Tang, and Jieping Ye. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4876–4883, 2020.

[62] Valentin Radu, Kuba Kaszyk, Yuan Wen, Jack Turner, Jose Cano, Elliot J. Crowley, Bjorn Franke, Amos Storkey, and Michael O'Boyle. Performance aware convolutional neural network channel pruning for embedded gpus, 2020.

[63] Adrian Holzbock, Achyut Hegde, Klaus Dietmayer, and Vasileios Belagiannis. Data-free backbone fine-tuning for pruned neural networks, 2023.

[64] Runyu Peng, Yunhua Zhou, Qipeng Guo, Yang Gao, Hang Yan, Xipeng Qiu, and Dahua Lin. Data-free weight compress and denoise for large language models, 2025.

[65] Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. A comprehensive survey of compression algorithms for language models, 2024.

[66] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, Chenggang Yan, and Yi Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE transactions on cybernetics*, 50(8):3594–3604, 2019.

[67] Hugo Tessier, Vincent Gripon, Mathieu Léonardon, Matthieu Arzel, Thomas Hannagan, and David Bertrand. Rethinking weight decay for efficient neural network pruning, 2022.

[68] Andrey Kuzmin, Markus Nagel, Mart van Baalen, Arash Behboodi, and Tijmen Blankevoort. Pruning vs quantization: Which is better?, 2024.

[69] Sebastian Cygert and Andrzej Czyżewski. Robustness in compressed neural networks for object detection, 2021.

[70] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 191–201, 2022.

[71] Haihao Shen, Naveen Mellempudi, Xin He, Qun Gao, Chang Wang, and Mengni Wang. Efficient post-training quantization with fp8 formats, 2024.

[72] Saurabhsingh Rajput and Tushar Sharma. Benchmarking emerging deep learning quantization methods for energy efficiency. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, pages 238–242. IEEE, 2024.

[73] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget?, 2021.

[74] Chinthaka Gamanayake, Lahiru Jayasinghe, Benny Ng, and Chau Yuen. Cluster pruning: An efficient filter pruning method for edge ai vision applications, 2020.

[75] Hsin-Pai Cheng, Yuanjun Huang, Xuyang Guo, Yifei Huang, Feng Yan, Hai Li, and Yiran Chen. Differentiable fine-grained quantization for deep neural network compression, 2018.

[76] Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization, 2021.

[77] Max Zimmer, Christoph Spiegel, and Sebastian Pokutta. Compression-aware training of neural networks using frank-wolfe, 2024.

[78] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2810–2819, 2019.

[79] Jun Nishikawa and Ryoji Ikegaya. Filter pre-pruning for improved fine-tuning of quantized deep neural networks, 2020.

[80] Sungmin Bae, Piotr Zielinski, and Satrajit Chatterjee. A closer look at hardware-friendly weight quantization, 2022.

[81] Micah Gorsline, James Smith, and Cory Merkel. On the adversarial robustness of quantized neural networks, 2021.

[82] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 14(6):1–50, 2023.

[83] Daria Cherniuk, Stanislav Abukhovich, Anh-Huy Phan, Ivan Oseledets, Andrzej Cichocki, and Julia Gusak. Quantization aware factorization for deep neural network compression, 2023.

[84] Yiren Zhao, Xitong Gao, Daniel Bates, Robert Mullins, and Cheng-Zhong Xu. Focused quantization for sparse cnns, 2019.

[85] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[86] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks, 2020.

[87] Quinten Van Baelen and Peter Karsmakers. Constraint guided model quantization of neural networks, 2024.

[88] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019.

[89] Shuhei Kashiwamura, Ayaka Sakata, and Masaaki Imaizumi. Effect of weight quantization on learning models by typical case analysis, 2024.

[90] Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet), 2022.

[91] Miguel A Carreira-Perpinán and Yerlan Idelbayev. “learning-compression” algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8532–8541, 2018.

[92] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization. *arXiv preprint arXiv:2103.06460*, 2021.

[93] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[94] Michela Paganini and Jessica Forde. Streamlining tensor and network pruning in pytorch, 2020.

[95] Jiawei Shao and Jun Zhang. Communication-computation trade-off in resource-constrained edge inference, 2020.

[96] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(5):2900–2919, 2023.

[97] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning?, 2020.

[98] Han Cai, Ji Lin, Yujun Lin, Zhijian Liu, Haotian Tang, Hanrui Wang, Ligeng Zhu, and Song Han. Enable deep learning on mobile devices: Methods, systems, and applications, 2022.

[99] Xiaolong Ma, Fu-Ming Guo, Wei Niu, Xue Lin, Jian Tang, Kaisheng Ma, Bin Ren, and Yanzhi Wang. Pconv: The missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices, 2020.

[100] Xue Geng, Zhe Wang, Chunyun Chen, Qing Xu, Kaixin Xu, Chao Jin, Manas Gupta, Xulei Yang, Zhenghua Chen, Mohamed M. Sabry Aly, Jie Lin, Min Wu, and Xiaoli Li. From algorithm to hardware: A survey on efficient and safe deployment of deep neural networks, 2024.

[101] Dwith Chenna. Evolution of convolutional neural network (cnn): Compute vs memory bandwidth for edge ai, 2023.

[102] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction, 2021.

[103] Zhangheng Li, Tianlong Chen, Linyi Li, Bo Li, and Zhangyang Wang. Can pruning improve certified robustness of neural networks?, 2022.

[104] Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Distributed pruning towards tiny neural networks in federated learning, 2023.

[105] Wenxiao Wang, Minghao Chen, Shuai Zhao, Long Chen, Jinming Hu, Haifeng Liu, Deng Cai, Xiaofei He, and Wei Liu. Accelerate cnns from three dimensions: A comprehensive pruning framework, 2021.

[106] Shaoyi Huang, Ning Liu, Yueying Liang, Hongwu Peng, Hongjia Li, Dongkuan Xu, Mimi Xie, and Caiwen Ding. An automatic and efficient bert pruning for edge ai systems, 2022.

[107] Artem Vysogorets and Julia Kempe. Connectivity matters: Neural network pruning through the lens of effective sparsity, 2023.

[108] Lizeth Gonzalez-Carabarin, Iris A. M. Huijben, Bastiaan S. Veeling, Alexandre Schmid, and Ruud J. G. van Sloun. Dynamic probabilistic pruning: A general framework for hardware-constrained pruning at different granularities, 2021.

[109] Gili Rosenberg, J. Kyle Brubaker, Martin J. A. Schuetz, Elton Yechao Zhu, Serdar Kadıoğlu, Sima E. Borujeni, and Helmut G. Katzgraber. Scalable iterative pruning of large language and vision models using block coordinate descent, 2024.

[110] Xin Qian and Diego Klabjan. A probabilistic approach to neural network pruning, 2021.

[111] Shibo Yao, Dantong Yu, and Ioannis Koutis. Neural network pruning as spectrum preserving process, 2023.

[112] Alexey Kruglov. Channel-wise pruning of neural networks with tapering resource constraint, 2018.

[113] Manas Gupta, Efe Camci, Vishandi Rudy Keneta, Abhishek Vaidyanathan, Ritwik Kanodia, Chuan-Sheng Foo, Wu Min, and Lin Jie. Is complexity required for neural network pruning? a case study on global magnitude pruning, 2024.

[114] Athanasios Glentis Georgoulakis, George Retsinas, and Petros Maragos. Feather: An elegant solution to effective dnn sparsification, 2023.

[115] Timothy Foldy-Porto, Yeshwanth Venkatesha, and Priyadarshini Panda. Activation density driven energy-efficient pruning in training, 2020.

[116] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey, 2023.

[117] Kamil Adamczewski, Yawei Li, and Luc van Gool. Shapley pruning for neural network compression, 2024.

[118] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 639–654. Springer, 2020.

[119] Chenyang Li, Jihoon Chung, Mengnan Du, Haimin Wang, Xianlian Zhou, and Bo Shen. On model compression for neural networks: Framework, algorithm, and convergence guarantee, 2024.

[120] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[121] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2009–2018, 2020.

[122] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[123] Jingfei Chang. Coarse and fine-grained automatic cropping deep convolutional neural network, 2020.

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

[125] Jamie McGowan, Wei Sheng Lai, Weibin Chen, Henry Aldridge, Jools Clarke, Jezabel Garcia, Rui Xia, Yilei Liang, Guillaume Hennequin, and Alberto Bernacchia. Efficient model compression techniques with fishleg, 2024.

[126] Tianyi Chen, Bo Ji, Yixin Shi, Tianyu Ding, Biyi Fang, Sheng Yi, and Xiao Tu. Neural network compression via sparse optimization, 2020.

[127] Deniz Gurevin, Shanglin Zhou, Lynn Pepin, Bingbing Li, Mikhail Bragin, Caiwen Ding, and Fei Miao. Enabling retrain-free deep neural network pruning using surrogate lagrangian relaxation, 2021.

[128] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.