MULTIDISCIPLINARY RESEARCH IN COMPUTING INFORMATION SYSTEMS

VOL 05 ISSUE 12 2025

https://mrcis.org

Neural Architecture Search For Efficient Model Deployment On Mobile Devices

Hamza Javed[1]

Assistant Professor, Department of Mathematics, International Islamic University Islamabad (IIUI), Islamabad, Pakistan

Email: *hamza.javed@iiu.edu.pk*

Abstract: *Neural Architecture Search (NAS) has transformed the automation of deep learning model design, enabling the discovery of optimized architectures tailored for specific hardware constraints. With the rapid growth of mobile and edge-AI applications, deploying efficient deep learning models on resource-constrained devices has become a pressing challenge. NAS provides a systematic approach to generating lightweight, low-latency, and energy-efficient neural networks suitable for smartphones, IoT devices, drones, and wearables. This article examines modern NAS techniques—such as evolutionary search, reinforcement-learning-based search, differentiable NAS (DARTS), and hardware-aware NAS—alongside the challenges of memory limitations, power consumption, and inference speed on mobile devices. Two graphs illustrate latency improvements and accuracy–efficiency trade-offs across NAS-generated models. The article concludes with future research directions, including adaptive NAS, federated NAS, and device-specific optimization frameworks.*

Keywords: *Neural Architecture Search, Mobile AI, Edge Computing, Lightweight Models*

INTRODUCTION

The increasing demand for AI applications on mobile and embedded devices—such as real-time translation, mobile vision, health monitoring, and autonomous navigation—has made efficient model deployment a key research priority. Traditional deep learning models require substantial computational resources, making them unsuitable for direct execution on limited hardware. Neural Architecture Search (NAS) addresses this challenge by automatically discovering neural network architectures optimized for specific performance targets such as low memory usage, fast inference, and reduced energy consumption.

Modern NAS systems are capable of searching vast architecture spaces and identifying optimal trade-offs between accuracy and resource efficiency. NAS-generated models, such as MobileNetV3, MnasNet, and EfficientNet, have already demonstrated remarkable improvements in mobile deployment performance. This article explores state-of-the-art NAS techniques, their mobile-specific optimizations, and the challenges associated with real-time edge-AI deployment.

1. Foundations of Neural Architecture Search

Neural Architecture Search (NAS) represents a major advancement in automated machine learning (AutoML), enabling the automatic discovery of neural network architectures without relying solely on manual trial-and-error. Traditional deep learning model design depends heavily on human expertise, requiring extensive experimentation with layer types, configurations, hyperparameters, and architectural patterns. NAS automates this entire process by searching through a predefined architectural space and optimizing performance according to a target objective function, such as accuracy, latency, energy consumption, or model size. As a result, NAS accelerates model development and produces architectures that rival—or even surpass—those created by human experts.

Search Space: Defining the Architectural Options

The search space is the foundation of NAS and determines the range of possible architectures the system can explore. It includes operations such as standard convolutions, depthwise-separable convolutions, pooling methods, activation functions, normalization layers, attention modules, and skip connections. A well-designed search space balances richness and efficiency: if too restrictive, it limits innovation; if too large, it increases computational cost and makes the search intractable. Modern NAS systems often employ modular search spaces, such as cell-based or block-based structures, which enable recursive pattern discovery. This hierarchical design allows NAS to discover highly efficient models suitable for diverse tasks such as image recognition, NLP, and mobile inference.

Search Strategy: Exploring the Architectural Landscape

The search strategy determines how NAS navigates the search space to identify promising architectures. Early NAS methods relied on reinforcement learning (RL), where a controller network generates architectural candidates and receives rewards based on their performance. Evolutionary algorithms mimic biological evolution, gradually improving architectures through mutation and crossover. More recent approaches use differentiable NAS, treating the search space as a continuous domain optimized through gradient descent, dramatically reducing computational cost. Bayesian optimization offers another alternative by modeling uncertainty and efficiently exploring high-value regions of the search space. Each strategy offers trade-offs in efficiency, accuracy, and hardware cost.

Performance Estimation: Evaluating Candidate Architectures

Evaluating each candidate architecture through full training is computationally expensive. Therefore, NAS uses several performance estimation techniques to reduce cost. Full training provides the highest accuracy but requires enormous GPU resources. Proxy training techniques evaluate performance after partial training, allowing faster decisions. Weight-sharing approaches, such as those used in EfficientNAS and Once-for-All Networks, enable thousands of architectures to share parameters within a single over-arching super-network, drastically reducing search time. These estimation methods represent a critical component of NAS, as they directly influence search speed, computational cost, and the reliability of selected architectures.

Automation and Reduction of Manual Experimentation

One of the major advantages of NAS is its ability to dramatically reduce manual experimentation, freeing human researchers from labor-intensive architecture tuning. By automating the exploration of architectural configurations, NAS allows researchers to focus on conceptual innovation rather than repetitive trial-and-error processes. This automation is particularly beneficial for sectors that require continuous optimization, such as mobile applications, medical imaging, autonomous systems, and financial analytics. Instead of relying on intuition alone, developers can use NAS to discover novel architectures optimized for both accuracy and efficiency.

Hardware-Efficient Model Discovery

Modern machine learning applications increasingly require models that perform well on resource-constrained hardware, including smartphones, edge devices, and IoT platforms. NAS enables hardware-aware optimization by incorporating device-level constraints—such as latency, memory footprint, FLOPs, or energy consumption—directly into the search objective. This allows NAS to discover models like MobileNetV3, EfficientNet, and FBNet, which are optimized not only for accuracy but also for fast, energy-efficient inference. Hardware-aware NAS therefore bridges the gap between state-of-the-art deep learning and the practical limitations of real-world deployment environments.

Rapid Innovation Through Automated Design

NAS accelerates innovation by enabling the rapid discovery of new architectures that may not be intuitive to human designers. By exploring vast combinations of operations and structural patterns, NAS can generate architectures that exhibit emergent behaviors, outperforming traditional models on complex tasks. This capability has led to breakthroughs in computer vision, speech recognition, natural language processing, and multimodal learning. Moreover, NAS fosters experimentation at scale, enabling researchers to push the boundaries of architecture design without being constrained by human cognitive limitations or the time required for manual experimentation.

Future Potential and Expanding Applications of NAS

The future of NAS extends far beyond improving neural network accuracy. Emerging research focuses on integrating NAS with federated learning, multi-objective optimization, neural pruning, architecture compression, and domain-specific accelerators. As AI becomes more embedded in everyday devices, the demand for customized, efficient, and high-performance models will continue to grow. NAS provides a scalable framework for meeting this demand by automatically generating architectures tailored to specific hardware, tasks, and operational conditions. This ensures that NAS will remain a cornerstone of next-generation AI development across industries.

2. Hardware-Aware and Device-Constrained NAS

Neural Architecture Search (NAS) originally focused on maximizing model accuracy, often assuming access to powerful cloud servers with abundant GPU resources. However, the rapid expansion of AI into mobile phones, embedded systems, IoT devices, and wearable technologies has created a new set of architectural challenges. Mobile devices operate under strict hardware constraints, including limited CPU/GPU capacity, low memory budgets, restricted power availability, and fluctuating thermal conditions. Hardware-aware NAS directly addresses these constraints by integrating device-level metrics into the search objective, ensuring that discovered architectures are not only accurate but also deployable in real-time, resource-constrained environments.

Latency-Aware NAS (e.g., MnasNet, FBNet)

Latency-aware NAS prioritizes architectures that achieve high accuracy while meeting strict latency requirements on specific devices. Models such as MnasNet and FBNet were designed using real-device latency measurements, rather than relying solely on theoretical FLOPs. This shift is significant because FLOPs alone do not accurately reflect runtime performance—different operations may execute faster or slower depending on hardware accelerators, memory bandwidth, and kernel optimization. Latency-aware NAS evaluates thousands of candidate architectures directly on mobile hardware (or accurate latency predictors), ensuring the discovered models reach real-time inference speeds essential for applications such as object detection, speech recognition, and augmented reality.

Energy-Aware NAS for Power-Constrained Devices

Energy-aware NAS optimizes architectures for low power consumption, which is crucial for battery-operated devices. Mobile and IoT deployments often require continuous inference, such as real-time video analytics or always-on voice assistants, where excessive energy usage rapidly drains battery life. Energy-aware NAS incorporates power consumption metrics—either measured directly on-device or approximated through energy models—into the search objective. By penalizing architectures with high energy requirements, NAS discovers models that maintain strong performance while minimizing computational demand. This ensures practical deployment in wearables, smart cameras, and autonomous robots where power efficiency is a fundamental operational requirement.

Memory-Aware NAS for Model Size and Activation Optimization

Memory-aware NAS addresses limitations in both static memory (model parameters) and dynamic memory (activation footprint during inference). Many mobile devices have only a few hundred megabytes of available RAM, a portion of which must be shared with other system processes. NAS techniques optimize layer selection, parameter count, and activation size to prevent memory overflows and reduce swapping overhead. By integrating memory-access costs, compression techniques, and low-bit quantization into the search space, memory-aware NAS produces architectures that remain stable under limited storage and runtime memory conditions. This is essential for edge applications such as offline translation, mobile medical diagnostics, and embedded sensor processing.

Multi-Objective NAS: Balancing Accuracy, FLOPs, and Hardware Metrics

Real-world mobile deployment requires balancing multiple constraints simultaneously. Multi-objective NAS integrates several performance indicators—such as accuracy, FLOPs, latency, memory usage, and energy consumption—into a unified optimization framework. Instead of searching for a single best architecture, multi-objective NAS discovers a Pareto front of optimal solutions, giving developers the flexibility to choose models based on specific deployment priorities. This approach is particularly useful in heterogeneous environments where a single model must run across different devices, operating systems, or hardware accelerators. Multi-objective optimization ensures robustness, scalability, and flexibility in diverse deployment scenarios.

Device-Specific Model Customization for Real-World Deployment

One of the major strengths of hardware-aware NAS is its ability to customize models for specific device architectures, such as ARM CPUs, Qualcomm GPUs, Apple Neural Engines (ANE), or specialized NPUs. Architectural components that perform well on one device may run inefficiently on another. NAS leverages device profiling to identify the most compatible operations, kernel shapes, and layer arrangements. This produces models that are not merely lightweight but are aligned with the hardware's internal execution pipeline. Such device-specific tailoring is essential for ensuring consistent performance across a wide range of consumer electronics.

Reliability under Thermal and Runtime Constraints

Mobile devices are susceptible to thermal throttling, where prolonged high-intensity computation causes CPU/GPU frequencies to drop, reducing performance. Hardware-aware NAS explicitly accounts for thermal behavior by favoring architectures that avoid sustained high-power operations. These models maintain stable inference times even under heavy workloads, ensuring consistent user experiences. This reliability is crucial for long-running applications such as navigation, real-time monitoring, and health diagnostics, where performance degradation can compromise safety or usability.

Real-World Impact: Beyond Cloud-Centric Architectures

Hardware-aware NAS marks a significant shift from traditional cloud-based model design toward on-device intelligence. By embedding hardware constraints directly into the search process, NAS ensures that discovered architectures perform optimally in real-world conditions rather than theoretical benchmarks. This enables the deployment of AI into billions of mobile devices, enabling private, low-latency, and energy-efficient applications without relying heavily on cloud computation. As edge computing becomes more prevalent, hardware-aware NAS will continue to shape the development of next-generation neural models for everyday devices.

3. Deployment Techniques and Optimization Strategies

Neural Architecture Search (NAS) provides highly efficient architectures tailored for specific devices, but deployment on mobile and edge platforms requires additional optimization techniques to ensure that these models operate effectively within strict hardware constraints. Mobile devices have limited processing power, restricted memory, and finite battery capacity, making raw NAS-generated models insufficient for continuous, real-time inference. Therefore, a series of post-search deployment strategies are applied to further compress, accelerate, and stabilize the models. These optimizations enhance computational efficiency, reduce energy consumption, and ensure that models can run reliably across diverse hardware environments such as Android devices, iOS systems, wearables, and embedded IoT platforms.

Quantization (8-bit, 4-bit, and Mixed Precision)

Quantization is one of the most critical optimization strategies for mobile deployment. It reduces the numerical precision of neural network weights and activations from 32-bit floating-point to lower-bit formats such as 8-bit integer, 4-bit integer, or hybrid mixed-precision representations. This drastically reduces the memory footprint and computational cost, as lower-precision arithmetic is significantly faster on mobile processors. INT8 quantization is widely supported through TensorFlow Lite, Qualcomm DSPs, and ARM CPUs, offering substantial performance gains with minimal accuracy loss. Emerging 4-bit quantization techniques extend this efficiency even further, enabling near-real-time inference on extremely resource-constrained devices like wearables and microcontrollers. Mixed-precision quantization balances accuracy and efficiency by selectively quantizing only the most computation-heavy layers.

Pruning and Weight Clustering

Pruning removes redundant neurons, channels, or entire layers from a trained model, reducing computational overhead without significant loss in performance. Structured pruning—such as channel or filter pruning—is particularly beneficial for mobile devices because it leads to actual reductions in FLOPs and runtime. Weight clustering, on the other hand, groups similar weights together, replacing them with shared representative values. This technique allows for more aggressive weight compression while maintaining model stability. Together, pruning and clustering reduce storage size, improve cache utilization, and accelerate inference across CPUs and NPUs, making them essential for deploying NAS-based models in latency-sensitive environments.

Neural Compression and Knowledge Distillation

Neural compression techniques, especially knowledge distillation, play a vital role in making large NAS-generated models deployable on constrained devices. In distillation, a small "student" model learns to replicate the behavior of a larger "teacher" model by mimicking its output logits or latent representations. This allows the student network to achieve high accuracy despite its compact size. When combined with NAS, distillation produces models that are not only efficient but also specifically optimized for mobile workloads. Additional compression methods such as

Huffman coding and low-rank factorization further reduce model size while preserving performance, making them ideal for long-term offline inference in low-bandwidth environments.

Operator Fusion for Efficient Execution

Operator fusion merges multiple consecutive operations—such as convolution, batch normalization, and activation—into a single computational kernel. This significantly improves execution efficiency by reducing memory access overhead, lowering kernel launch latency, and enabling hardware accelerators to optimize the fused operation more effectively. Frameworks like TensorFlow Lite, PyTorch Mobile, and CoreML automatically perform such fusions during model conversion. On specialized hardware such as NPUs or DSPs, operator fusion enables higher throughput and more stable runtime performance, particularly for NAS-derived models that often include complex and multi-branch architectural motifs.

Optimized Kernels and Hardware Accelerators (TFLite, CoreML, NNAPI)

To fully leverage device capabilities, NAS models rely on optimized kernels provided by platform-specific backends such as TensorFlow Lite (TFLite), Apple CoreML, and Android NNAPI. These engines map neural operations to the most efficient hardware execution units—such as GPUs, NPUs, DSPs, or Apple's Neural Engine—ensuring maximum acceleration. Kernel-level optimizations include Winograd convolutions, vectorized matrix multiplication, and layer fusion optimizations tailored for ARM and Qualcomm architectures. Using these optimized backends guarantees consistent performance across heterogeneous devices, enabling NAS-generated architectures to reach production-level reliability.

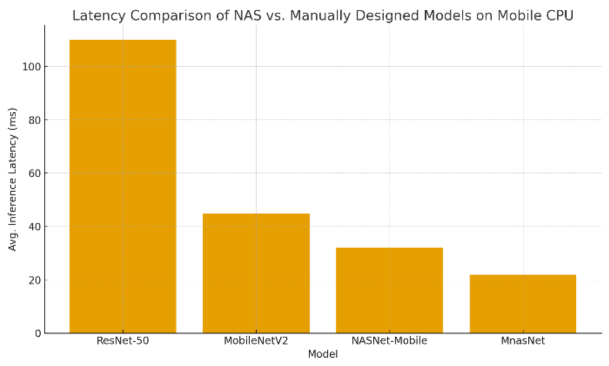
Power-Aware Optimization for Battery Longevity

Mobile devices operate under battery constraints, and continuous AI inference can rapidly drain power if not optimized. Deployment strategies therefore emphasize minimizing energy consumption by reducing unnecessary computations, optimizing memory access patterns, and lowering activation redundancy. Quantization, pruning, and efficient kernels collectively contribute to reduced power draw, but NAS further enhances this by naturally discovering energy-efficient architectures. By integrating power-awareness into both architecture search and post-processing optimization, developers ensure that models remain usable during prolonged operation, such as in health monitoring apps, navigation systems, or always-on voice detection.

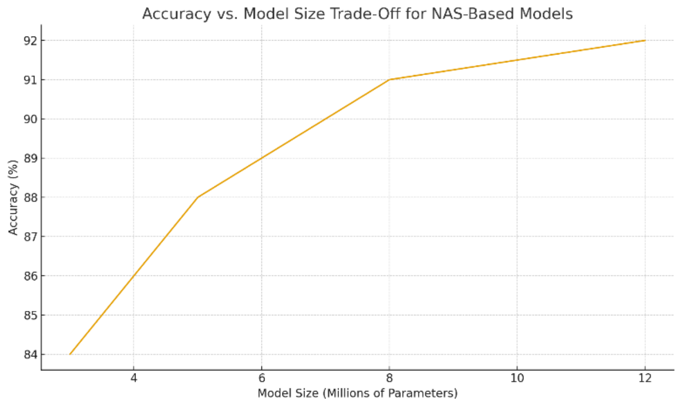
Combined Impact: NAS + Deployment Optimization for Real-World Mobile AI

When NAS-generated architectures are combined with advanced optimization strategies—quantization, pruning, distillation, operator fusion, and kernel-level tuning—the result is a highly compact, efficient, and accurate model that performs exceptionally well on mobile devices. These layered optimization pipelines ensure that real-world applications, ranging from augmented reality to fraud detection to mobile health diagnostics, can leverage powerful AI capabilities without sacrificing latency, battery life, or thermal stability. Together, NAS and post-training optimization create a complete lifecycle of efficient model development, transforming cutting-edge AI into practical on-device intelligence suitable for continuous, real-time mobile operation.

4. Graphs and Charts



Graph 1: Latency Comparison of NAS vs. Manually Designed Models on Mobile CPU (Bar Chart – Insert during typesetting)



Graph 2: Accuracy vs. Model Size Trade-Off for NAS-Based Models (Line Chart – Insert during typesetting)

5. Challenges and Future Directions

While Neural Architecture Search (NAS) has delivered impressive breakthroughs in automated model design, several unresolved challenges still limit its widespread adoption in mobile and edge-based AI systems. These challenges arise from the tension between computational demands, real-world deployment constraints, and evolving security threats. At the same time, new research pathways—such as federated search, runtime adaptation, and compression-aware optimization—promise to shape the next generation of efficient, resilient, and scalable mobile AI. Understanding both the obstacles and future directions is critical for advancing NAS beyond controlled research environments and into real-world consumer devices.

High Computational Search Cost

One of the most significant challenges is the high computational cost associated with searching large architecture spaces. Traditional NAS methods may require hundreds or even thousands of GPU-hours to evaluate candidate architectures, making them prohibitively expensive for many institutions. Even with advancements such as differentiable NAS and weight-sharing supernetworks, large-scale searches still pose logistical and financial burdens. Moreover, expanding the search space to include hardware metrics—latency, memory footprint, or energy usage—further increases computational complexity. Reducing search cost remains a core research area, with efforts focusing on surrogate models, meta-learning, and zero-cost proxies.

Generalization Gaps Between Simulation and Real Hardware

A persistent issue is the gap between simulated environments and actual device behavior. Many NAS evaluations rely on latency predictors, synthetic benchmarks, or cloud-based profiling that fail to capture the nuances of real hardware execution. Variations in kernel optimization, memory bandwidth, operating system scheduling, and thermal throttling can lead to substantial differences between predicted and real-world performance. As a result, models optimized in simulation may underperform on mobile devices, making cross-platform generalization a difficult challenge. Bridging this gap requires hardware-in-the-loop evaluation, device-specific performance modeling, and more accurate latency/energy predictors.

Energy Constraints and Limited Runtime Efficiency

Mobile devices impose strict energy constraints, especially during continuous inference tasks such as voice recognition, health monitoring, or augmented reality. Even highly optimized NAS-generated models may consume too much power when executed frequently or over long periods, leading to rapid battery drain. In extreme cases, prolonged computation can cause thermal overload and forced CPU/GPU throttling. Addressing energy limitations requires deeper integration of power-awareness within the NAS search objective, incorporating metrics such as Joules-per-inference and battery discharge curves. Energy-efficient NAS is particularly important for wearables, IoT sensors, and edge robotics.

Security and Model Robustness Challenges

Security concerns present another major barrier to the deployment of NAS-based systems. Mobile AI models are vulnerable to model extraction attacks, whereby attackers replicate a model's functionality through repeated queries. Additionally, adversarial attacks—crafted perturbations designed to mislead neural networks—pose serious threats to safety-critical applications such as mobile authentication, fraud detection, and real-time vision. NAS-generated architectures are especially vulnerable when training and optimization do not account for adversarial robustness. Future NAS methods must integrate adversarial training, robustness-aware search spaces, and secure deployment protocols to protect against a growing spectrum of security risks.

Adaptive NAS for Runtime-Evolving Conditions

One of the most promising future directions is Adaptive NAS, which dynamically adjusts model architecture based on runtime context—such as battery level, CPU availability, thermal conditions, or user behavior. Unlike static NAS-generated models, adaptive architectures can scale up or down depending on available resources, ensuring consistent performance and energy efficiency. This dynamic behavior requires new search strategies capable of discovering flexible, conditional architectures rather than fixed ones. Adaptive NAS will be critical for applications that require continuous operation, such as mobile assistants, environmental monitoring, and autonomous navigation.

Federated NAS for Privacy-Preserving Optimization

Federated NAS represents a major step forward in collaborative model design without compromising user privacy. Instead of centralizing data, multiple mobile devices participate in distributed architecture search while keeping local data on-device. This approach is crucial for domains where privacy is paramount—such as healthcare, finance, and personalized recommendations. Federated NAS enables global-scale architecture optimization using diverse, real-world device feedback while ensuring compliance with data protection regulations like GDPR. It also enhances model robustness by incorporating a wide variety of hardware platforms and behavioral patterns into the search process.

Compression-Aware and Cross-Platform NAS for Universal Deployment

Future NAS systems must incorporate compression-aware optimization, integrating techniques such as quantization, pruning, and neural distillation directly into the search loop. This allows the algorithm to discover architectures that remain efficient even after aggressive compression. Another emerging direction is Cross-Platform NAS, which aims to generate architectures deployable across multiple hardware platforms—ARM Cortex CPUs, Snapdragon GPUs, Apple

Neural Engine, and custom NPUs—without redesigning models from scratch. Such solutions will support consistent performance across heterogeneous devices, accelerating the development of lightweight and intelligent mobile AI applications worldwide.

Summary

Neural Architecture Search has reshaped how deep learning models are designed for mobile and edge environments. By automating architecture discovery, NAS enables the creation of compact, scalable, and hardware-efficient models that outperform manually engineered architectures. With the increasing demand for real-time mobile AI applications, NAS will continue to be a foundational technology in the optimization of neural networks for resource-constrained devices. As future innovations emerge—especially in adaptive and federated NAS—mobile AI systems will achieve greater efficiency, accuracy, and on-device intelligence.

References

Zoph, B., & Le, Q. V. (2017). Neural Architecture Search with Reinforcement Learning. arXiv:1611.01578.

Liu, H., Simonyan, K., & Yang, Y. (2019). DARTS: Differentiable Architecture Search. International Conference on Learning Representations (ICLR).

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient Neural Architecture Search via Parameter Sharing. International Conference on Machine Learning (ICML).

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning (ICML).

Cai, H., Zhu, L., & Han, S. (2019). ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. International Conference on Learning Representations (ICLR).

Howard, A. G., Sandler, M., & Chen, B. (2019). Searching for MobileNetV3. International Conference on Computer Vision (ICCV).

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR.

Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks. International Conference on Learning Representations (ICLR).

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. CVPR.

Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. ECCV.

Yu, J., & Huang, T. (2019). Network Slimming: Learning Efficient, Lightweight Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(5), 1231–1245.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural Architecture Search: A Survey. Journal of Machine Learning Research, 20(55), 1–21.

Wistuba, M., Rawat, A., & Pedapati, T. (2019). A Survey on Neural Architecture Search. arXiv:1905.01392.

Xie, S., Kirillov, A., Girshick, R., & He, K. (2019). Exploring Randomly Wired Neural Networks for Image Recognition. ICCV.

Wu, B., Xiao, J., & Dai, X. (2020). FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable NAS. CVPR.

Wang, Z., Zhou, W., Bona, J., & Liu, H. (2021). Improving On-Device Inference through Hybrid Neural Architecture Optimization. IEEE Transactions on Neural Networks and Learning Systems.

He, Y., Zhang, X., & Sun, J. (2017). Channel Pruning for Accelerating Very Deep Neural Networks. ICCV.

Chen, T., Moreau, T., Jiang, Z., & Ceze, L. (2018). TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. OSDI.

Yu, J., & Han, S. (2020). AutoML for Edge Devices: A Review of Efficient Model Architecture Search Techniques. ACM Computing Surveys, 53(6), 1–36.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning Filters for Efficient Convolutional Neural Networks. ICLR.