Journal of Electronic Research and Application, 2026, Volume 10, Issue 4
http://ojs.bbwpublisher.com/index.php/JERA
Online ISSN: 2208-3510
Print ISSN: 2208-3502

Optimization and Implementation of a Lightweight Neural Network Architectures for Edge Computing

Xianke Meng, Shi Xiong
Nanjing Research Institute of Electronic Engineering, Nanjing 210023, Jiangsu, China

Abstract: For the resource constraints and real-time requirements of edge computing scenarios, this paper systematically studies the optimization and implementation method of a lightweight neural network architecture. From the constraints of edge device computing power, power consumption constraints and the diversity of deployment environment, this paper discusses the architecture design strategy based on convolution decomposition, efficient embedding of attention mechanism and width depth collaborative optimization, and then proposes a comprehensive optimization scheme of training perception quantization, pruning and neural architecture search combined optimization, reasoning engine collaborative adaptation and end-to-end dynamic optimization. Through the deep integration of algorithm innovation and hardware characteristics, a complete technical framework from model design to edge deployment is constructed, aiming to achieve a good balance between accuracy and efficiency under resource constraints, and provide theoretical support and practical guidance for the model selection and deployment optimization of edge intelligent system.

Keywords: Edge calculation; Lightweight neural network; Architecture optimization; Model compression

Online publication: May 21, 2026

1. Introduction

With the deep integration of Internet of things and artificial intelligence technology, edge computing scenarios pose a serious challenge to the deployment of neural network models. Cloud reasoning is limited by network bandwidth and delay, which is difficult to meet the needs of real-time sensitive applications such as autonomous driving and industrial control. It is an inevitable trend to move intelligent computing forward to edge devices. However, edge devices are generally faced with practical bottlenecks such as limited computing resources, strict power consumption constraints, and complex deployment environment. Traditional high-precision neural networks cannot directly run on such platforms due to large number of parameters and intensive computing [1]. Therefore, the research on the optimization and implementation of lightweight neural network architecture for edge computing has important theoretical value and application significance. Starting from the actual constraints of edge computing scenarios, this paper systematically

discusses the lightweight architecture design method, model compression optimization strategy and end-to-end deployment implementation technology, aiming to build a complete technical framework from model design to edge deployment, and provide theoretical support and practical guidance for efficient and stable edge intelligent applications in resource constrained environments.

2. Constraint analysis of edge computing scene on neural network model

2.1. Computing resources and power consumption limits of edge devices

Edge devices are usually composed of embedded processors, mobile terminals or IOT nodes. Their computing power is far lower than that of cloud servers, mainly reflected in the limited number of computing cores, low dominant frequency and limited memory bandwidth. At the same time, such devices often rely on battery power supply or energy collection, and power consumption has become a key constraint to the continuous work of the system. If the structure of the neural network model is too deep and the parameters are too large, it will directly lead to the increase of calculation delay and the sharp rise of energy consumption, and even exceed the power consumption range of equipment thermal design, resulting in frequency reduction or system instability [2]. Therefore, in the model design stage, we must take the amount of computation and storage access as the core optimization goal, and realize the balance between reasoning efficiency and energy consumption under limited resources through lightweight operation, hardware friendly structure and memory reuse mechanism, so as to meet the basic premise of edge deployment.

2.2. Real time requirements and delay sensitivity characteristics

Edge computing scenarios are widely involved in automatic driving, industrial control, intelligent security and other applications that have strict requirements on response time, and the reasoning delay often needs to be controlled in milliseconds or even microseconds. Different from the cloud reasoning relying on network transmission, the edge end eliminates the communication overhead through local computing, but the calculation delay of the model itself is still the key bottleneck of the real-time performance of the system. Delay sensitivity shows that the end-to-end time from data acquisition to result output must meet the threshold constraint of closed-loop control or interactive experience, and any overrun may lead to system failure or decline in user experience [3]. Therefore, the lightweight neural network architecture needs to establish a fine trade-off between accuracy and delay, and ensure that the reasoning delay in the worst case can still meet the real-time requirements of the application scenario by means of operator fusion, pipeline parallel and early exit mechanism.

2.3. Requirements of diversity of deployment environment on model robustness

The edge device deployment environment is highly heterogeneous and dynamic, covering different hardware platforms, a variety of sensor inputs and complex working conditions. To address these challenges and improve model robustness, several deployment-related factors and mitigation strategies are summarized in **Table 1**. The same model may need to run on arm, DSP, NPU and other architectures. The differences between hardware instruction set and memory access characteristics pose challenges to the compatibility and performance consistency of the model [4]. In addition, edge scenes often face non ideal conditions such as input data distribution offset, illumination change, occlusion interference, etc., which require the model to have good robustness and generalization ability to avoid significant decline in accuracy due to environmental

changes. Therefore, in the process of lightweight architecture design, it is not only necessary to consider the static hardware adaptation, but also need to introduce strategies such as data enhancement and domain adaptation to improve the stability and reliability of the model in the diversified deployment environment.

Table 1. Challenges, impacts, and mitigation strategies for model robustness in diverse edge deployment environments

Deployment challenge	Potential impact on model	Example mitigation strategy
Hardware heterogeneity	Inconsistent performance across ARM, DSP, NPU	Cross-platform profiling and lightweight optimization
Sensor input variability	Reduced accuracy with different sensor types or resolutions	Sensor-aware normalization and multi-modal training
Dynamic environmental conditions	Model drift under changing illumination, occlusion, or noise	Domain adaptation, robust data augmentation, online fine-tuning

3. Lightweight neural network architecture design method

3.1. Structure compression strategy based on convolution decomposition

The core of lightweight neural network architecture design is to reduce the amount of calculation and parameters of the model from the structural level. Convolution decomposition is the basic means to achieve this goal. The traditional standard convolution performs spatial feature extraction and channel information fusion at the same time, and the computational complexity is a product of the convolution kernel size and the number of input and output channels. In view of this feature, deep separable convolution decomposes it into two independent steps: deep convolution and pointwise convolution. The former is responsible for feature extraction in spatial dimension, and the latter is responsible for cross-channel information fusion. The theoretical calculation can be reduced to about one ninth of the standard convolution, becoming the key component of mobilenet, xception and other series models [5]. On this basis, variant forms such as packet convolution, channel shuffling, hole convolution are further derived. By controlling the number of packets and expansion rate of convolution operation, the computational overhead is further reduced while maintaining the ability of feature expression. In addition, the structural reparameterization technology achieves an effective balance between accuracy and efficiency by using a multi branch topology in the training phase and converting it into a single path convolution in the reasoning phase. These structural compression strategies together constitute the basic methodology of lightweight architecture design, and provide a stable structural basis for subsequent attention mechanism embedding and collaborative optimization.

3.2. Efficient embedding of attention mechanism in lightweight architecture

Attention mechanism enables the model to focus on the key information in the input by dynamically adjusting the feature weight, which has a significant effect on improving the representation ability, but the additional computational overhead and memory access introduced by it may weaken the actual benefits of the lightweight model on edge devices [6]. Therefore, the efficient embedding of attention mechanism has become an important issue in lightweight architecture design. The channel attention mechanism represented by squeeze and exception recalibrates the channel through global pooling and lightweight gating structure. The number of parameters accounts for only a small proportion of the total number of models, but it can bring considerable accuracy improvement. On this basis, coordattention and other improved schemes further

embed the location information into the channel attention, so that the model can capture the important features of space and channel dimensions at the same time without increasing the complexity. For mobile terminal reasoning scenarios, lightweight attention modules usually use point by point convolution to replace the full connection layer, or reduce the parameters of gating structure through grouping strategy to ensure that the module itself has high hardware friendly characteristics [7]. At the same time, the attention mechanism and the aforementioned convolution decomposition strategy can form a collaborative design. For example, the attention module is placed between deep convolution and point-by-point convolution to strengthen the effectiveness of feature transfer under the condition of low computational cost, so that the lightweight network can still maintain strong feature discrimination ability when the parameter budget is limited, so as to achieve a higher balance between accuracy and efficiency in edge deployment.

3.3. Model width and depth collaborative optimization mechanism

The performance of lightweight neural network depends not only on the design of a single convolution layer, but also on the collaborative configuration between the overall width and depth of the network. The width corresponds to the number of channels in each layer, which determines the representation capacity and calculation density of the model; Depth corresponds to the number of layers of the network, which affects the level of feature abstraction and the size of receptive fields [8]. Simply increasing the width will significantly increase the amount of calculation and memory occupation, while simply deepening the network may introduce the gradient propagation problem and delay accumulation, which need to be balanced under the unified optimization goal. The introduction of neural architecture search provides a systematic means for such collaborative optimization. By defining multi-dimensional search space including width, depth, convolution type and so on, the optimal structure combination is automatically explored with the help of reinforcement learning or evolutionary algorithm to maximize the accuracy of the model within the specified calculation budget. In addition, the method based on composite scaling adjusts the width, depth and input resolution proportionally through a unified scaling factor to maintain the structural consistency of the model under different resource constraints and avoid the efficiency loss caused by empirical stacking [9]. In the actual design process, the shallow network usually adopts a wider channel setting to retain enough low-level information, while the deep network moderately compresses the width to control the total amount of calculation, forming a pyramid type capacity allocation mode. This collaborative optimization mechanism of width and depth enables the lightweight architecture to achieve better accuracy and delay balance under limited resources, laying a structural foundation for efficient deployment at the edge.

4. Model optimization strategy and edge deployment implementation

4.1. Block-based training perceptual quantization and mixed precision compression

By reducing the numerical accuracy of weights and activation values, model quantization can significantly compress the volume of the model and improve the reasoning speed at the edge. However, direct quantization often introduces accuracy loss, especially for lightweight structures. Training perceptual quantization first simulates the quantization error in the training process, and adjusts the model weight adaptively through the pseudo quantization node, which effectively reduces the accuracy gap between the quantized model and the floating-point model. In addition, a block-based quantization strategy is introduced, where input data is segmented into multiple sub-blocks and quantization operations are performed independently, significantly

reducing quantization errors caused by outliers and improving post-quantization model accuracy [10]. On this basis, hybrid precision compression further distinguishes the sensitivity of each layer to quantization, retains high-precision calculations for key layers, and uses low-precision representation for redundant layers to achieve a fine balance between precision and compression ratio. According to the hardware characteristics of edge devices, the quantization strategy also needs to be aligned with the computing unit of the target platform. For example, 8-bit symmetric quantization is used on npus that support symmetric quantization to maximize the speedup, while floating-point 16 or dynamic quantization is used on microcontrollers that do not support integer operation. The advantages of the quantified model in memory occupation and memory access bandwidth are particularly prominent, which can make the model that could not be deployed run smoothly in resource constrained devices, and become the key technical means of landing at the edge (**Figure 1**).

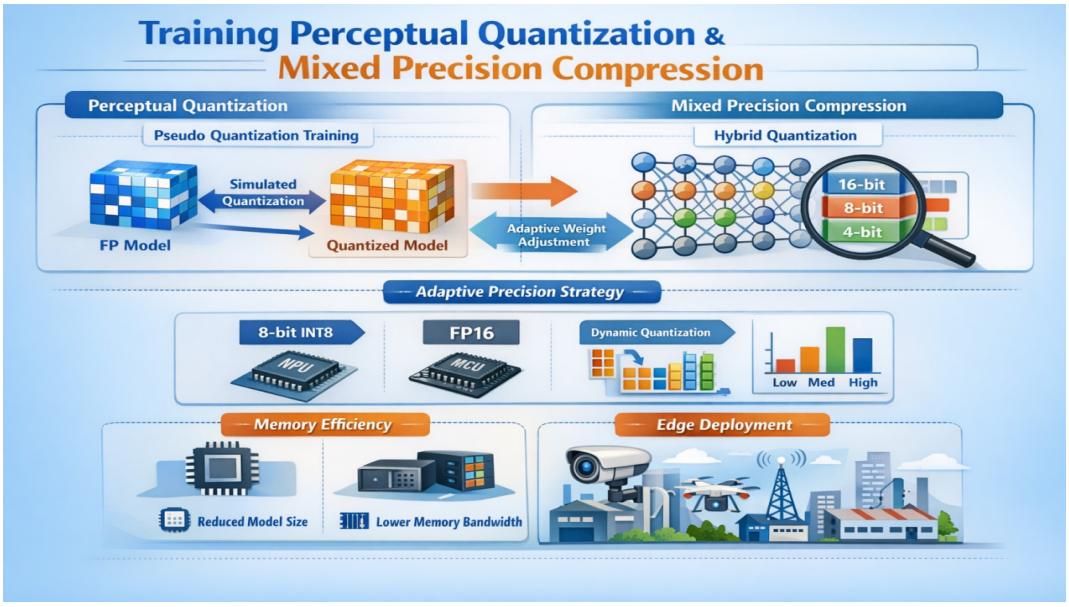

Figure 1. Mechanism framework of training perceptual quantization and mixed-precision compression for edge deployment.

4.2. Joint optimization of pruning and neural architecture search

Pruning reduces the amount of model parameters and computation by removing redundant connections or convolution kernels, but its effect is highly dependent on the selection of initial structure and pruning strategy. The traditional pruning method usually aims at the pre training model, which has the problem that the compression potential is limited due to the fixed structure. Neural architecture search directly generates a lightweight structure that meets resource constraints from the search space, and advances the compression idea of pruning to the architecture design stage. The joint optimization of the two forms a closed loop: the architecture search generates an efficient initial structure, and pruning further eliminates redundancy on this basis. At the same time, the sparsity after pruning can be included in the evaluation index in the search process, leading the architecture to evolve in the direction of easy compression. The coordination of structural pruning and search strategy is particularly critical. The former ensures hardware acceleration friendliness by pruning by channel or layer, while the latter provides global optimal structural configuration. The jointly optimized model is superior to the single technology path in accuracy maintenance, compression

rate and reasoning speed, and provides an integrated and efficient solution for the scenes with strictly limited resources at the edge.

4.3. Collaborative adaptation of hardware acceleration library and reasoning engine

The actual reasoning performance of the lightweight model at the edge depends not only on the structure of the model itself, but also on the adaptation of the underlying accelerator library to the reasoning engine. The hardware acceleration library is deeply optimized for the instruction set and microarchitecture of a specific processor, such as cmsis-nn of arm platform, snpe of Qualcomm platform and the supporting runtime of various NPU manufacturers. The computing density is maximized through technologies such as handwritten assembly, operator fusion and memory reuse. The inference engine is responsible for model parsing, graph optimization and hardware scheduling. In tflite, MNN, ncnn and other frameworks, graph optimization methods such as operator fusion, constant folding, memory reuse are used to reduce runtime overhead. The key of collaborative adaptation lies in the matching between the model structure and the acceleration library characteristics. For example, the acceleration effect of deep separable convolution on some npus is limited, and the structure needs to be adjusted to avoid performance degradation. At the same time, the reasoning engine should support heterogeneous computing scheduling, and reasonably allocate different operators in the model to CPU, NPU, DSP and other computing units to minimize the overall delay. The three-layer cooperation of the algorithm, operator library and reasoning engine enables the lightweight model to fully release the hardware potential.

5. Conclusion

This paper focuses on the optimization and implementation of lightweight neural network architecture for edge computing. Starting from the resource constraint analysis of edge scenarios, the key constraints of computing power, power consumption, real-time and deployment diversity on model design are clarified. At the level of architecture design, the core methods such as convolution decomposition, efficient embedding of attention mechanism and collaborative optimization of width and depth are discussed, and the hardware friendly lightweight structure foundation is constructed. On this basis, a comprehensive optimization strategy combining training perception quantization, pruning and neural architecture search joint optimization, reasoning engine collaborative adaptation and end-to-side dynamic optimization is further proposed, forming a complete technology chain from model design to edge deployment. The research shows that the effective implementation of lightweight neural network depends on the deep integration of algorithm innovation and hardware characteristics. Through the multi-level cooperation of structure compression, accuracy optimization and runtime adaptation, it can achieve a good balance between accuracy and efficiency in limited resources. The technical framework combed in this paper can provide reference for model selection and deployment optimization of edge intelligent system, and promote the large-scale application of lightweight neural network in resource constrained scenarios.

Disclosure statement

The author declares no conflict of interest.

References

[1] Cai T, 2025, Design of a Lightweight Intrusion Detection Algorithm for Smart Factories Oriented to Industrial Edge Computing. Cybersecurity and Informatization, 2025(6): 127–129.

[2] Lin C, Yan W, Ji G, et al., 2024, A Survey of Lightweight Methods for Deep Neural Network Parameters. Journal of China Academy of Electronics and Information Technology, 19(4): 350–363+379.

[3] He W, Liu Z, Liu H, et al., 2024, Electromagnetic Field Simulation Method for Transformers Based on a Lightweight Hybrid-Gate Neural Network. Power System Technology, 48(5): 2143–2151.

[4] Lu M, 2021, Research on Lightweight Models under Edge Computing. Modern Computer, 27(25): 31–35.

[5] Ma F, Wang B, Dong X, et al., 2020, Power Vision Edge Intelligence: Deep Visual Acceleration Technology Driven by Edge Computing in Power Systems. Power System Technology, 44(6): 2020–2029.

[6] Liao F, Liu B, Huang J, et al., 2024, Research on Cloud-Edge Collaborative Application Mode of Intelligent Distribution Station Rooms Based on Lightweight Models. Automation & Instrumentation, 2024(3): 210–215.

[7] Xu X, Zhou C, Hu Z, et al., 2024, A Survey on Lightweight Deep Neural Network Models for Edge Intelligence. Computer Science, 51(7): 257–271.

[8] Pang S, 2025, Design of Lightweight Optimization Algorithms for System Integration in Edge Computing Scenarios. Product Reliability Report, 2025(7): 213–214.

[9] Zhang K, Lu Z, Nie T, et al., 2023, Analysis of Intelligent Edge Computing Software Technologies for Unmanned Equipment. Acta Armamentarii, 44(9): 2611–2621.

[10] Ling X, Jiang J, Bao J, 2025, An Optimization Method for Quantization-Aware Training inEdge Environments. Journal of China Academy of Electronics and Information Technology, 20(5): 465–474.