SYNERGY PUBLISHING PTE. LTD.

Modern Electronic Technology

http://www.advmodoncolres.sg/index.php/amor/index

Research on Lightweight Convolutional Neural Network Algorithm Deployment and Performance Enhancement for Embedded Devices

LI Jinchao LI Sheng

School of Electronic Information, Xijing University, Xi'an, Shaanxi, 710123

ARTICLE INFO

Article history
Received: 12 March 2025
Accepted: 24 March 2025
Published Online: 30 March 2025

Keywords:
lightweight convolutional neural network
embedded device
model compression
parameter quantization
performance optimization

ABSTRACT

In the context of the rapid development of intelligent terminals and the Internet of Things (IoT), Convolutional Neural Networks (CNNs) have achieved remarkable results in fields such as image recognition, object detection, and semantic segmentation. However, traditional deep networks are characterized by large structures, complex computations, and high energy consumption, making them difficult to deploy directly on resource-constrained platforms such as embedded devices. This paper systematically analyzes the core principles, optimization strategies, and deployment methods of lightweight CNNs, focusing on techniques such as model compression, network pruning, parameter quantization, and knowledge distillation. Combined with the concept of software–hardware co-optimization, it explores efficient mapping and performance improvement paths of models on embedded platforms. Through experimental analysis of representative lightweight networks (e.g., MobileNet, ShuffleNet, GhostNet), it is verified that model parameters and computation can be significantly reduced while maintaining accuracy. The study shows that the co-optimization of hardware characteristics and algorithm structures is the key direction for achieving high performance and low power consumption balance in intelligent inference on embedded devices.

Introduction

The rise of deep learning has driven rapid advances in artificial intelligence (AI) in the domains of perception and cognition, with CNNs demonstrating strong feature extraction and representation capabilities in image recognition, speech recognition, and natural language processing. However, traditional CNNs such as VGG, ResNet, and DenseNet often sacrifice computational efficiency and storage overhead in pursuit of accuracy. Their massive parameter scales and high computational complexity make them difficult to deploy directly in resource-constrained embedded systems. With the emergence of edge computing and intelligent IoT, model lightweighting and efficient deployment have become hot research topics in deep learning. This paper, based on the characteristics of embedded devices, systematically explores algorithm design, optimization mechanisms, and deployment practices of lightweight CNNs, aiming to provide low-latency, low-power algorithmic solutions that maintain performance for embedded intelligent systems.

I. Theoretical Foundations and Development of Lightweight Convolutional Neural Networks

(1) Basic Concepts of Lightweight Network Design

The core objective of lightweight CNNs is to achieve

**Corresponding Author:*
LI Sheng
Email: sheng@mail.xjtu.edu.cn

DOI: https://doi.org/10.26549/met.v9i1.36160

performance comparable to large models with fewer parameters and computations. The main ideas include reducing convolutional kernel size, decreasing channel dimensions, and adopting separable convolution structures. Depthwise Separable Convolution decomposes standard convolution into depthwise and pointwise convolutions, significantly reducing computational cost; Group Convolution reduces parameter connections through grouping operations; Channel Shuffle enhances information flow efficiency. These structural innovations effectively reduce network load and lay the foundation for lightweight network design.

(2) Evolution of Representative Lightweight Network Architectures

Since the introduction of MobileNet, lightweight networks have evolved from structural sparsification to adaptive optimization. The MobileNet series, centered on depthwise separable convolutions, flexibly controls model complexity through width and resolution multipliers; ShuffleNet, based on grouped convolutions, improves feature expression via channel shuffling; GhostNet reduces computation by generating redundant features through linear transformations; EfficientNet achieves coordinated optimization of width, depth, and resolution through compound scaling. These architectures not only improve computational efficiency but also provide mature frameworks for embedded deployment.

(3) Application Scenarios of Lightweight Networks

In embedded intelligent terminals, lightweight CNNs are widely used in mobile visual recognition, real-time object detection, pose estimation, and scene understanding. Smart cameras, drones, wearable devices, and industrial inspection systems all require a balance between computational efficiency and energy constraints. Lightweight networks enable efficient operation on platforms such as ARM, DSP, and FPGA, providing critical support for edge-side intelligence and laying the foundation for "end–edge–cloud" collaborative decision-making architectures.

II. Core Optimization Techniques of Lightweight Convolutional Neural Networks

(1) Model Pruning and Structural Sparsification

Model pruning reduces model complexity by removing redundant parameters and connections and is an essential approach to lightweighting. Pruning can be classified into weight pruning and structural pruning. The former selects parameters based on importance metrics, while the latter removes channels or entire layers. Recently, adaptive pruning algorithms based on gradient sensitivity and loss constraints have achieved significant compression without notably degrading accuracy. Structural sparsification further utilizes sparse matrix multiplication to optimize computation, enabling efficient execution of computation-intensive tasks on embedded devices.

(2) Parameter Quantization and Low-Bit Computation

Parameter quantization maps floating-point weights to low-bit fixed-point representations, significantly reducing storage and computation costs. Common methods include fixed-point quantization, dynamic quantization, and mixed-precision quantization. Eight-bit integer quantization is widely used on mobile platforms, while 4-bit and binary networks (Binary Neural Networks) achieve high compression in specific tasks. To mitigate performance degradation caused by quantization errors, researchers have proposed Quantization-Aware Training (QAT), which simulates quantization during training to maintain accuracy stability. Combined with SIMD instruction set optimization on hardware, quantized networks can achieve inference performance comparable to full-precision models.

(3) Knowledge Distillation and Multi-Task Collaborative Optimization

Knowledge distillation transfers knowledge from a large teacher model to a smaller student model, balancing accuracy and efficiency. The distillation process can occur at the feature, output, or intermediate layers, aligning feature distributions or class probabilities to realize knowledge transfer. Combined with multi-task learning frameworks, lightweight networks can maintain generalization while improving robustness. Furthermore, structure re-parameterization based on distillation allows networks to adopt different configurations during training and inference, dynamically balancing performance and efficiency.

III. Algorithm Deployment and Performance Optimization Strategies on Embedded Devices

(1) Software–Hardware Co-Optimization Pathways

In embedded systems, algorithm performance depends not only on model structure but also on hardware resource allocation and execution mechanisms. Software–hardware co-optimization achieves performance maximization through coordinated design across the model, compilation, and hardware layers. At the model layer, op-

 DOI: https://doi.org/10.26549/met.v9i1.36160

timization focuses on pruning and parallel restructuring; the compilation layer reduces data transfer overhead via graph optimization, operator fusion, and memory reuse; the hardware layer exploits heterogeneous architectures (e.g., CPU+NPU or FPGA accelerators) to offload computation-intensive modules. This hierarchical synergy effectively improves inference speed and reduces energy consumption.

(2) Model Adaptation for Specific Platforms

Different embedded platforms possess distinct instruction sets, cache mechanisms, and memory bandwidths. For ARM Cortex processors, NEON instructions can accelerate matrix operations; FPGA platforms can employ pipelined designs and on-chip memory optimization to minimize latency; NVIDIA Jetson GPU platforms utilize TensorRT for structural rearrangement and precision tuning. These platform-specific adaptations are key to ensuring efficient lightweight network operation.

(3) Deployment Frameworks and Toolchain Support

With the widespread adoption of deep learning, mainstream AI frameworks such as TensorFlow Lite, PyTorch Mobile, NCNN, MNN, and OpenVINO provide deployment interfaces for embedded devices. These tools support model conversion, graph optimization, and automatic quantization, integrating closely with hardware vendor SDKs. Through end-to-end optimization, developers can rapidly migrate models from training to inference. Experiments show that deploying MobileNetV3 using TFLite or NCNN on Raspberry Pi achieves approximately threefold inference acceleration and 40% power reduction.

IV. Comprehensive Research on Performance Enhancement of Lightweight Algorithms

(1) Multi-Level Fusion Optimization Mechanism

To address the constraints of limited computational power, memory, and energy in embedded devices, researchers have developed multi-level fusion optimization mechanisms to dynamically balance network structure and hardware resources. This mechanism jointly optimizes network design, operator implementation, and runtime scheduling, reducing redundant computation at the structural level while employing adaptive pruning and dynamic quantization at the algorithmic level, allowing networks to adjust automatically according to real-time workloads and energy budgets. At runtime, Dynamic Inference strategies select computational paths based on input complexity, reducing unnecessary computation through early exit or branch inference. These mechanisms significantly lower latency and energy consumption without sacrificing accuracy, achieving efficient coordination among precision, speed, and power efficiency.

(2) Neural Architecture Search and Automated Deployment

Neural Architecture Search (NAS) has transformed network design from an expert-driven to an algorithm-driven process. By leveraging reinforcement learning, evolutionary algorithms, or gradient-based methods, NAS explores vast architectural spaces to find optimal topologies under limited computational budgets. In embedded contexts, Hardware-Aware NAS incorporates device characteristics as optimization constraints to achieve optimal latency and power balance on target platforms. With the maturity of automated toolchains, NAS can now integrate with compilers and deployment frameworks, enabling end-to-end optimization from architecture generation to quantized hardware deployment. This process shortens the model design cycle and enhances portability and consistency across heterogeneous platforms, supporting large-scale deployment of lightweight algorithms on embedded devices.

(3) Energy Modeling and Inference Acceleration Mechanisms

In embedded environments, power control is a crucial performance metric. To address energy bottlenecks caused by high-frequency computation and frequent memory access, researchers have developed hierarchical energy modeling frameworks spanning from the operator to the system level. These models predict power consumption across network layers, providing guidance for structure optimization at the design stage. By analyzing data reuse patterns, reducing external memory access, and improving on-chip cache hit rates, energy consumption can be significantly reduced. Operator-level parallelization, pipelined scheduling, and data reordering further enhance throughput while maintaining accuracy. Experimental results show that such optimizations improve overall inference speed by about 30% and reduce energy consumption by 20%, effectively alleviating performance bottlenecks and enabling low-power, high-efficiency AI deployment in embedded systems.

V. Conclusion

The deployment and optimization of lightweight CNNs on embedded devices are crucial for achieving universal access to intelligent computing. Through multi-level al-

DOI: https://doi.org/10.26549/met.v9i1.36160

gorithmic optimizations such as pruning, quantization, distillation, and neural architecture search, inference speed and energy efficiency can be significantly improved without compromising accuracy. Future research will focus on deeper integration between algorithms and hardware through software–hardware co-design, energy-aware scheduling, and dynamic inference mechanisms to build adaptive intelligent models. As AI chips and edge computing architectures continue to mature, lightweight deep learning for embedded systems will become the driving force behind the migration of AI from cloud to edge, providing robust support for efficient applications in IoT, intelligent manufacturing, and smart transportation.

Project

This work was supported by National Natural Science Foundation of China(No.11974289).

References

[1] Xiao J ,Xu L ,Li C , et al.Lightweight visible damage detection algorithm for embedded systems applied to pipeline automation equipment[J].Journal of Pipeline Science and Engineering,2025,5(2):100254-100254.

[2] Anver R S ,Deepambika A V ,Rahiman A M , et al.Emotional Speech Generation: An Approach Using Convolutional Neural Networks (CNN) Based Generative Adversarial Network[J].Circuits, Systems, and Signal Processing,2025,44(11):1-23.

[3] Murti A M ,Setianingsih C ,Kusumawardhani E , et al.Cedarwood Quality Classification using SVM Classifier and Convolutional Neural Network (CNN) [J].International Journal of Advanced Computer Science and Applications (IJACSA),2022,13(11):

[4] Kavita B ,Vijaya M .Evaluation of CNN model by comparing with convolutional autoencoder and deep neural network for crop classification on hyperspectral imagery[J].Geocarto International,2020,37(3):1-15.

[5] Mousavi S M ,Rahmani F .Improving the detection efficiency of IRan ANtineutrino Detector (IRAND) based on Convolutional Neural Network (CNN)[J].Nuclear Engineering and Technology,2026,58(5):104118-104118.

 DOI: https://doi.org/10.26549/met.v9i1.36160