A Survey of Efficient CNN Hardware Acceleration Technologies for Edge Computing

Rui Cao [1] * and Jinrun Tian [2]

[1] School of Information and Communication, Guilin University of Electronic Technology, Guilin, 541004, China
[2] School of Information and Communication Engineering, Zhongyuan University of Technology, Zhengzhou, 451191, China
rayvoncr@mails.guet.edu.cn

Abstract. With the rapid development of edge computing, the deployment of convolutional neural networks (CNNs) on edge devices has become increasingly necessary to meet the demand for intelligent applications. However, edge devices are often constrained by limited computing resources, storage capacity, and strict power consumption requirements. These challenges make the design of efficient CNN models and advanced hardware acceleration technologies critical research areas. This paper focuses on these two aspects and provides an overview. Firstly, it introduces model compression techniques to reduce the computational complexity and storage of CNNs. Both structured pruning, which removes entire filters or layers, and unstructured pruning, which eliminates individual weights, are discussed. Lightweight network architectures, such as the MobileNet series, are also highlighted as efficient solutions for balancing performance and resource use. Secondly, the paper examines CNN hardware acceleration technologies, emphasizing platforms like Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), which enhance performance and energy efficiency. Finally, this paper summarizes the current progress in these fields and discusses future research directions, focusing on balancing accuracy, efficiency, and hardware constraints to optimize CNN deployment on edge devices.

Keywords: Edge computing, convolutional neural networks (CNNS), hardware acceleration, model compression.

1 Introduction

With the rapid development of Internet of Things (IoT) and artificial intelligence (AI) technologies, traditional cloud computing architectures face many challenges when processing massive amounts of data. The traditional cloud computing model needs to upload massive data to the cloud server, but because the cloud server is far from the terminal device, problems such as transmission rate, energy loss, response delay, network interference, and data security are difficult to avoid in the transmission process [1]. The concept of edge computing can be traced back to the content delivery network

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_78

(CDN) proposed by Akamai in 1998, but the first appearance of the term "edge computing", It was in a 2013 internal report by Ryan LAMOTHE at the Pacific Northwest National Laboratory [2].

Edge computing is closer to the data source, and data storage and computing tasks can be carried out on edge computing nodes, reducing the intermediate data transmission process, emphasizing proximity to users, providing users with better intelligent services, thereby improving data transmission performance, ensuring real-time processing, and reducing latency [3]. This approach enables low-latency service responses. Model compression and quantization techniques demonstrate significant potential for enhancing efficiency in practical applications. This research provides a new theoretical basis and method for deep learning model optimization in edge computing by systematically discussing the compression and quantization techniques of CNN model. Secondly, these techniques significantly reduce the computational complexity and storage requirements of CNN models. This improvement enhances the efficiency of edge device operations and meets the demands for real-time and low-latency processing.

2 Efficient CNN model design

2.1 Model Compression Technique

Model pruning is a simple way to pre-train models by eliminating unimportant parameters or structures. The aim of this technique is to decrease the model size while minimizing any negative effects on its performance. Pruning methods can be categorized as unstructured and structured pruning [4]. Unstructured pruning cuts down on computation by eradicating unimportant weights from the neural network. There are multiple important steps involved in the unstructured pruning process. The first step is to train an initial model. Then, the importance of each parameter is assessed based on the gradient magnitude. In the second step, eliminate unimportant parameters and pruning is finished. To restore its performance, the pruned model is trained. Finally, the model is re-evaluated, pruned, and fine-tuned until it achieves satisfactory performance and compression by repeating the steps above.

While unstructured pruning can achieve a high compression rate of the model, it usually results in an unstructured sparse matrix because of the limitations of the underlying hardware and computing library. After pruning, the model may not utilize all of its hardware resources during the actual calculation process, and the calculation speed may not meet the expected level [4]. By removing structured components in the neural network, like entire neurons, channels, layers, or filters, structured pruning can reduce computational effort and number of parameters in the model. The unit of structured pruning is the filter or channel, which still preserves the original convolutional structure. It also produces a regular structured model that can help with hardware acceleration. Further, pruning the filter of the convolutional layer also reduces the number of input channels corresponding to the next layer. This regular structure is very effective for compressing and accelerating the network model [5].

This approach proves particularly advantageous for practical applications requiring efficient computation. However, its effect is highly dependent on the adaptability of the hardware architecture and pruning strategy. Improper pruning may lead to irreversible loss of accuracy, require repeated fine-tuning and evaluation, and may face additional optimization costs and resource consumption during actual deployment. In general, structured pruning has hardware-friendly features that make it better for practical scenarios, while unstructured pruning can achieve higher compression rates for theoretical scenarios. Table 1 provides a detailed comparison of the compression rates and hardware adaptability of both methods. The aspects of both are summarized in Table 1.

Table 1. Comparison of the characteristics of structured pruning and unstructured pruning.

Characteristics	Structured pruning	Unstructured pruning
Pruning Granularity	Coarse-grained (channels, neurons, layers)	Fine-grained (individual weights)
Compression Rate	Lower	Higher
Hardware Friendliness	Higher	Lower
Applicable Scenarios	Practical deployment (e.g., mobile devices)	Theoretical scenarios

As deep learning is applied in more and more fields, model pruning will become a key technology for efficient, lightweight deployments.

2.2 Lightweight network architecture design

Lightweight network architecture design is a significant research direction in deep learning. Its role is to design efficient and compact neural network models that enable high-performance reasoning on resource-constrained devices such as mobile devices and embedded systems. Here is a classic example of MobileNet, which illustrates the principles of lightweight network architecture in practical applications. In 2017, Google introduced MobileNet for both mobile and embedded devices. High efficiency and a straightforward network architecture are two benefits of MobileNet [6]. The convolutional network structure of MobileNet is shown in Fig. 1.

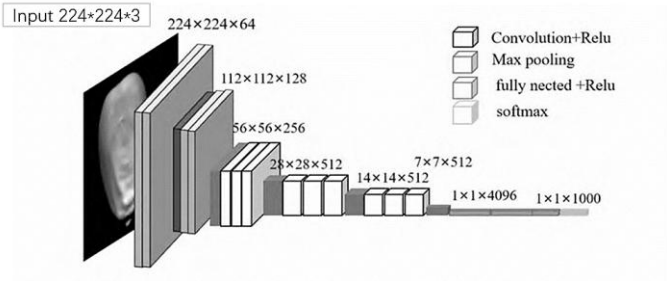

Fig. 1. The convolutional network structure of MobileNet [6].

Researchers at Google have launched MobileNetV1. It is designed for mobile and embedded vision applications, with the advantage of providing speed and accuracy for this application. Using depthwise separable convolution is the core innovation of MobileNet V1. This design reduces the calculation cost and model size on the basis of traditional CNN. The standard convolution layer is divided into two layers by depth separable convolution, one is a depthwise convolution and the other is a pointwise convolution [7]. MobileNet V1 laid the foundation for MobileNet V2 and MobileNet V3 because of its efficient design on mobile.

MobileNet V2 introduced the concepts of Inverted Residual Block and Linear Bottleneck on the basis of MobileNet V1. On the basis of ensuring higher accuracy of the model, the innovation of this design reduces the computational burden [7]. The efficiency enhancement of MobileNet V2 has a significant impact on edge computing and has become a classic lightweight network. MobileNet V3 is the pinnacle of the MobileNet series. It integrates advanced technologies such as NAS, h-swish activation function, and SE module to enhance performance while ensuring computational efficiency. MobileNet V3's efficiency and high performance are crucial for physical applications like image classification [7]. As a milestone in lightweight neural network architecture, the technology introduced by MobileNetV3 has had a profound impact on the field of deep learning, especially the deployment of models for mobile and embedded devices, and it also has had a significant impact on subsequent model development.

3 A CNN Hardware Accelerator for Edge Computing

3.1 FPGA-based Accelerator

Architectural design. The data flow architecture allows data to flow between processing units in a specific order to improve processing efficiency and parallelism. The pulsating array architecture has many processing units forming an array, and the data propagates in it like a wave, which can process large-scale data efficiently. It can expand the cycle structure, reduce the cycle, control the cost, and improve the running speed. Pipeline technology divides the processing process into multiple stages and processes different data in parallel at each stage to improve the overall throughput. Data reuse reduces the number of accesses and reduces power consumption and latency by reusing read data [8].

Typical case analysis. The application of Gemmini accelerator on FPGA, through modifying Gemmini to deploy Yolov7 model on Xilinx ZCU102 FPGA development board to achieve real-time performance level. Gemmini has made configuration adjustments such as increasing the number of PE, two-way data flow, increasing buffer capacity, and using AutoTVM for automatic tuning and DSP packaging technology optimization to improve energy efficiency.

Based on the computational requirements of the target task, the original number of PE in Gemmini was increased from 8 to 16. In this way, when processing image data,

more convolution operations and other tasks can be processed in parallel, which greatly improves the computational efficiency [9]. For example, when processing convolution operations for high-resolution images, more PE can calculate different image regions at the same time, reducing the overall processing time. The data path is reformed to realize two-way data flow. For example, between the convolution layer and the pooled layer, data can not only flow from the convolution layer to the pooled layer, but also part of the results of the pooled layer can flow back to the convolution layer for further processing or correction when necessary. In this way, the data can be processed more flexibly, especially when dealing with some neural network structures with feedback mechanisms, and the two-way data flow can better adapt to the needs of the model. Increased the buffer size from 16KB to 32KB. When dealing with large image datasets, larger buffers can store more intermediate data, reducing the number of times data is read from external memory. For example, in the continuous multi-layer convolution operation, the intermediate result can be temporarily stored in the buffer for subsequent layers to use directly, avoiding frequent access to the slow external memory and reducing the data access delay. According to the hardware architecture of Gemmini and the above configuration adjustment, AutoTVM is used to carry out the neural network model in image recognition task. AutoTVM automatically searches for different scheduling policies and parameter combinations. For instance, it explores various block sizes and loop expansions during convolutional layer calculations. Through extensive experiments and evaluations, an optimal set of parameter configurations was identified, leading to a 5% increase in accuracy and a 30% improvement in reasoning speed on the improved Gemmini.

3.2 Accelerator based on ASIC

TPU (Tensor Processing Unit) has a large number of matrix multiplication units and dedicated storage structures, which can efficiently handle matrix operations in deep learning [10]. NPU (Neural Network Processing Unit) is designed for neural networks, with specialized neuron processing units and synaptic storage units, which can process a large number of neuronal calculations in parallel.

Optimization technology. A customized instruction set can include special instructions tailored to CNN calculations to enhance operational efficiency. Data accuracy optimization Reduces storage and computation without affecting performance by reducing data accuracy.

Reconfigurable CNN Acceleration for Edge Devices. The NPU architecture proposed by the Samsung team leverages input-output channel parallelism and neural network sparsity to accelerate operations. However, it is not ideal for resource-constrained edge devices. The reconfigurable CNN coprocessor for edge computing, based on the data flow mode processed by channel, proposes a two-level distributed storage scheme and a flexible local memory access mechanism, which improves the flexibility of hardware architecture.

3.3 Other Hardware acceleration Platforms

GPU low power optimization for edge devices. By optimizing process technology and designing efficient power management modules, power consumption can be reduced. Additionally, task scheduling and resource allocation improvements make the GPU more energy efficient during CNN task processing.

DSP instruction set expansion for CNN calculation. Add special instructions for CNN core operations such as convolutional operation and pooling operation, improve the ability of DSP to process CNN calculation, optimize the order of instruction execution and data access mode, and improve the operation efficiency.

3.4 New Computing architecture

In-memory computing integrates storage and computing functions to reduce data handling and improve energy efficiency. This approach, alongside brain-like computing, represents a significant advancement in CNN acceleration. Brain-like computing emulates the structure of human neurons and synapses. It offers distinct advantages in handling complex perceptual and pattern recognition tasks, making it suitable for CNN acceleration in edge computing

4 CONCLUSION

This paper provides a comprehensive overview of efficient CNN model designs and hardware accelerators, with a focus on techniques such as model pruning and lightweight network architectures. It highlights the importance of reducing computational complexity through model compression techniques—such as structured and unstructured pruning—and lightweight architectures like the MobileNet family, which are specifically designed to address the resource constraints of edge computing environments. Additionally, the paper explores hardware accelerators based on FPGA and ASIC technologies, emphasizing the optimization strategies and structural designs that can significantly enhance the computational efficiency of CNNs on edge devices. Through typical use cases, this study illustrates the significant potential of these technologies in edge computing applications.

Despite these advancements, challenges remain. For instance, while model pruning and other compression techniques effectively reduce computational and storage requirements, they often result in a loss of model accuracy. Developing methods to achieve substantial compression while maintaining high accuracy remains an open area of research. Similarly, FPGA-based accelerators, though highly flexible and adaptable to various workloads, often lag behind ASICs in terms of performance. On the other hand, ASIC-based accelerators provide superior performance due to their specialization but lack the versatility to adapt to rapidly evolving algorithmic demands. This trade-off highlights the critical need to strike a balance between flexibility and performance.

In conclusion, while significant progress has been made in both efficient model design and hardware acceleration for CNNs, there remain important research questions to address. Future efforts should focus on developing innovative solutions that optimize accuracy, flexibility, and performance, ensuring that CNNs can be deployed more effectively in resource-constrained edge computing scenarios. This will be essential for unlocking the full potential of edge intelligence in real-world applications.

Acknowledgments (authors contribution). All the authors contributed equally and their names were listed in alphabetical order.

References

1. Zhang, Y., Liang, Y., Yin, M., Zhang, X.: 'Survey on the Methods of Computation Offloading in Mobile Edge Computing', Chinese J. Comput., **44**(12), 2406-2430 (2021)
2. Lei, B., Zhao, Q., Zhao, H.: 'Overview of Edge Computing and Computing Power Network', ZTE Technol. J., **27**(3), 3-6 (2021)
3. Cao, K., Liu, Y., Meng, G., Tu, Q.: 'An Overview on Edge Computing Research', IEEE Access, **8**, 85714-85728 (2020)
4. Zeng, H., Hu, H., Lin, X., Wang, R.: 'Deep Neural Network Compression and Acceleration: An Overview', J. Signal Process., **38**(1), 183-194 (2022)
5. Huo, Z., Zheng, Y., Chen, Y.: 'Research progress of model lightweight and acceleration', J. Signal Process., **38**(3), 35-40 (2023)
6. Cao, Y., Ma, D.: 'Portable corn kernel image recognition device based on lightweight MobileNet', Mod. Agric. Mach., **1**, 72-74 (2025)
7. Dantas, P.V., da Silva, W.S., Cordeiro, L.C., Haeusler, E.H.: 'A comprehensive review of model compression techniques in machine learning', Appl. Intell., **54**, 11804-11844 (2024)
8. Li, B., Qin, G., Zhu, S.: 'Design of FPGA Accelerator Architecture for Convolutional Neural Network', J. Front. Comput. Sci. Technol., **14**(3), 437-448 (2020)
9. Peccia, F.N., Pavlitska, S., Fleck, T., Bringmann, O.: 'Efficient Edge AI: Deploying Convolutional Neural Networks on FPGA with the Gemmini Accelerator', Proc. Int. Conf. Digital System Design, Paris, France, 418-426 (2024)
10. Hu, Y., Liu, Y., Liu, Z.: 'A survey on convolutional neural network accelerators: GPU, FPGA and ASIC', Proc. Int. Conf. Computer Research and Development, Shenzhen, China, 100-107 (2022)