Proceedings of CONF-CDS 2025 Symposium: Application of Machine Learning in Engineering
DOI: 10.54254/2755-2721/2025.AST26509

Lightweight Network Architecture — Optimization of Deep Learning Models

Zhaoyi Mao

College of Engineering, Shanghai Ocean University, Shanghai, China
2961164801@qq.com

Abstract. With the widespread application of deep learning in image classification tasks, the demand for lightweight convolutional neural networks (CNNs) has been steadily increasing in resource-constrained environments such as mobile devices and embedded systems. To balance classification accuracy and computational efficiency, numerous lightweight network architectures—such as MobileNet, ShuffleNet, SqueezeNet, and EfficientNet—have introduced various techniques, including depthwise separable convolution, inverted residual blocks, channel shuffle, Fire modules, and compound scaling. This paper proposes an innovative lightweight network architecture, LwNet, which integrates and customizes modules based on these classical structures to enable more efficient feature extraction and information transmission. Specifically, we design a module named LightBlock, which combines 1×1 convolution expansion, depthwise separable convolution, grouped convolution, and channel shuffle operations. The module also incorporates residual connections and attention mechanisms to further enhance model performance. Experimental results demonstrate that LwNet achieves excellent classification accuracy on the CIFAR-10 dataset, offering a superior balance between accuracy and efficiency compared to other mainstream lightweight networks. This study provides a novel perspective for designing lightweight neural networks and offers reproducible code and experimental data to support further research in related fields.

Keywords: Lightweight Convolution, MobileNet, ShuffleNet, SqueezeNet

1. Introduction

With the rapid development of deep learning in the field of computer vision, convolutional neural networks (CNNs) have been widely applied to tasks such as image classification, object detection, and semantic segmentation, achieving remarkable results. In particular, the introduction of deep neural networks such as AlexNet [1], VGG [2], and ResNet [3] has led to groundbreaking progress on various large-scale datasets. However, these networks often involve a massive number of parameters and high computational complexity, making them difficult to deploy efficiently in resource-constrained environments such as embedded systems, mobile devices, and edge computing platforms. As a result, designing lightweight convolutional neural networks (Lightweight CNNs) has become a hot research topic in recent years. To reduce model size and computational cost while maintaining accuracy, researchers have proposed various innovative lightweight network

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.AST26509

architectures. Models such as the MobileNet series [4], ShuffleNet [5], and SqueezeNet [6] have significantly improved computational efficiency by introducing techniques like depthwise separable convolution, group convolution, channel shuffle, and Fire modules. Additionally, EfficientNet [7] introduced a compound scaling strategy that simultaneously scales network depth, width, and input resolution, further enhancing both efficiency and accuracy. However, most existing lightweight networks tend to focus on a single optimization strategy, often leading to a trade-off between accuracy and computational efficiency. Therefore, achieving a better balance between accuracy and efficiency by effectively combining different lightweight techniques remains a key challenge.To address this, we propose a new lightweight convolutional neural network architecture that integrates multiple lightweight design concepts—such as depthwise separable convolution, group convolution, channel shuffle, and inverted residual structures—into a custom-designed **LightBlock** module. This integration enables the network to achieve an improved trade-off between performance and efficiency.

The main contributions of this paper are as follows: Structural Innovation: A modular structure named LightBlock is proposed, which integrates multiple lightweight strategies to achieve efficient information extraction and feature representation. Lightweight Design: While maintaining a compact network size, the design effectively incorporates key techniques such as expansion convolution, depthwise separable convolution, group convolution, channel shuffle, and residual connections. Experimental Validation: Extensive experiments are conducted on the CIFAR-10 dataset to demonstrate the advantages of the proposed network in terms of accuracy and parameter size, along with comparative analysis against several classical models. The remainder of this paper is organized as follows: Section 2 reviews related work and recent progress in lightweight models. Section 3 provides a detailed description of the proposed network architecture and module design. Section 4 introduces the experimental setup and evaluation metrics, and presents comparative results. Section 5 concludes the paper and outlines directions for future research.

2. Related work

2.1. Depthwise separable convolution and inverted residual structure

The MobileNet series is one of the representative architectures of lightweight CNNs. MobileNetV1 [4] first introduced Depthwise Separable Convolution, which decomposes standard convolution into two parts: depthwise convolution and pointwise convolution. This significantly reduces computational complexity and the number of parameters. By adopting this strategy, MobileNetV1 greatly decreases FLOPs and enhances computational efficiency on resource-constrained devices. MobileNetV2 [5] builds upon this by introducing the Inverted Residual Block and Linear Bottleneck. This structure expands the number of channels before compressing them, while preserving residual connections across layers, thereby improving feature reuse and training stability. Further advancements, such as MobileNetV3 [6], incorporate Neural Architecture Search (NAS) and Squeeze-and-Excitation modules, which further enhance the model's lightweight nature and expressive capacity.

2.2. Group convolution and channel shuffle

Another notable and widely recognized lightweight design is the ShuffleNet series, particularly ShuffleNetV1 [8] and ShuffleNetV2 [9], which have significantly contributed to the development of efficient neural networks. ShuffleNetV1 introduced two fundamental concepts: group convolution

DOI: 10.54254/2755-2721/2025.AST26509

and the channel shuffle mechanism. Group convolution helps reduce computational cost by splitting the input channels into smaller groups, allowing convolutions to be applied independently within each group. However, this separation could limit information flow between groups. To solve this issue, the channel shuffle operation was proposed, which rearranges the channels within each group, enabling the exchange of features across different groups. This cross-group feature exchange effectively prevents information bottlenecks and leads to improved model performance. Building upon the success of ShuffleNetV1, ShuffleNetV2 further refined the architecture by optimizing the entire computation path, considering hardware-related factors such as memory access costs, parallel processing efficiency, and inference latency. This hardware-aware optimization resulted in a more efficient and scalable design. The improved design of ShuffleNetV2 makes it particularly suitable for applications on mobile devices and embedded systems, where low power consumption and high computational performance are essential, offering significant advantages in real-time, high-performance scenarios.

2.3. Efficient scaling and compression module

In addition to the aforementioned convolutional structure designs, SqueezeNet [10] and EfficientNet [11] proposed highly innovative compression and scaling strategies that further revolutionized lightweight neural network architectures. SqueezeNet achieves remarkable model compression through the Fire module [12], which extensively utilizes 1×1 convolutions to significantly reduce feature dimensions, followed by a smaller number of computationally less expensive 3×3 convolutions for spatial feature extraction. This efficient design allows SqueezeNet to drastically reduce the model size to just 1/50th of that of AlexNet, all while maintaining a comparable level of accuracy, significantly lowering the storage and memory requirements. This compression strategy makes SqueezeNet particularly suitable for applications in resource-limited environments such as mobile devices and embedded systems [13]. On the other hand, EfficientNet introduced the groundbreaking compound scaling strategy, which simultaneously adjusts multiple aspects of the network architecture, including its depth, width, and input resolution. This method allows the model to collaboratively optimize these parameters, thus enhancing both accuracy and computational efficiency without overwhelming the system. EfficientNet demonstrated state-of-the-art performance across multiple benchmark datasets, providing valuable insights into scalable network design. Notably, its compound scaling strategy not only improves model accuracy but also enhances the network's adaptability, making it more suitable for deployment across various hardware platforms and computational resources, from edge devices to high-performance GPUs [14]. This strategy has significantly expanded the potential applications of lightweight models, providing greater flexibility and efficiency in real-world scenarios [15].

3. Medthod

To significantly reduce model parameters and computational overhead while maintaining high classification accuracy, this paper proposes a lightweight convolutional neural network architecture that is both structurally flexible and efficiently integrated. The overall model incorporates the advantages of various classic modules, including depthwise separable convolutions, inverted residual connections, channel shuffle mechanisms, and 1×1 compression/expansion strategies. The following sections will present the proposed method in terms of module design, overall network architecture, training details, and implementation workflow.

DOI: 10.54254/2755-2721/2025.AST26509

3.1. Module design: LightBlock

To enhance the network's representational capacity while reducing redundant computations, this paper proposes a fundamental unit that integrates multiple lightweight design principles—LightBlock. This module incorporates the following key techniques: 1. 1×1 Expansion Convolution (Expansion Layer): A 1×1 convolution is first applied to expand the number of input channels to r times the original (where r is the expansion ratio), enhancing the non-linear representational power. This operation is computationally inexpensive and helps increase feature dimensionality. 2. 3×3 Depthwise Convolution: Depthwise convolution is performed independently on each channel to extract spatial features. This operation significantly reduces computational cost compared to standard convolutions and is inspired by MobileNetV1. 3. Grouped Convolution + Channel Shuffle:Group convolution limits inter-channel connections to specific groups, while the channel shuffle operation rearranges feature channels to promote cross-group information flow. This design follows the principles of ShuffleNetV2. 4. 1×1 Projection Convolution (Projection Layer): A final 1×1 convolution reduces the channel dimension to the desired output size. Batch Normalization is applied to stabilize training. If the input and output channels are equal, a residual connection is added, similar to the inverted residual structure in MobileNetV2. Each sub-module is followed by a ReLU activation function to further enhance the network's non-linear feature expression. The overall structure of LightBlock balances representational power and computational efficiency, making it an ideal unit for building deep lightweight networks.

3.2. Network architecture design (OurNet)

Based on the LightBlock module, this paper constructs a compact image classification network—OurNet, designed for small- and medium-scale image classification tasks such as CIFAR-10. The network architecture consists of the following components: 1. Stem (Initial Convolution Layer): A standard 3×3 convolution is used to map the input image to a 32-dimensional feature map, followed by Batch Normalization (BN) and ReLU activation for initial feature extraction. 2. Feature Extraction Module (LightBlock Stack): Multiple LightBlocks are stacked to form a deep network structure. The specific configuration is as follows:Layer 1: Input channels: 32, Output channels: 32, Expansion ratio: 4, Number of groups: 2; Layer 2: Input channels: 32, Output channels: 64, Expansion ratio: 4, Number of groups: 4; Layer 3: Input channels: 64, Output channels: 64, Expansion ratio: 4, Number of groups: 4; ReLU and BN are used between modules, and some blocks include residual connections to enhance information flow and gradient propagation. 3. Classification Module (Head): At the end of the network, global average pooling (GAP) is applied to compress spatial features into a vector representation, which is then passed through a fully connected layer to produce the final class prediction. The total parameter count of the network is approximately 2.8M, which is significantly smaller than mainstream models like ResNet18 and MobileNetV2. This design achieves efficient inference with low resource consumption while maintaining competitive accuracy.

3.3. Training settings and experimental procedure

To validate the performance of the proposed network on standard image classification tasks, systematic and comprehensive experiments were conducted on the CIFAR-10 dataset, a well-established benchmark for evaluating image classification models. The training procedure and settings are as follows: Data Augmentation and Preprocessing: PyTorch's built-in data augmentation

DOI: 10.54254/2755-2721/2025.AST26509

functions were employed, which include random cropping (with padding=4), random horizontal flipping, and normalization (using the mean and standard deviation values from the ImageNet dataset). These strategies are designed to effectively enhance the model's ability to generalize to new, unseen data. Training Configuration: The optimizer used was Adam, with an initial learning rate of 0.001, and the CrossEntropyLoss function was utilized to minimize classification error. The batch size was set to 128 for training and 100 for testing, with a total of 50 epochs. The model was deployed on both CPU and GPU for adaptive computation, ensuring flexibility across different hardware platforms. Model Evaluation Metric: The primary evaluation metric for the model's performance is classification accuracy on the test set. During the training process, loss curves and accuracy trends were plotted to observe the model's convergence behavior and to ensure that it was not overfitting. Reproducibility Statement: The model is fully implemented using PyTorch, and the code covers the entire network architecture, training pipeline, and evaluation scripts. Complete structural definitions, along with training examples, are provided to facilitate reproducibility and to ensure that future researchers can build upon this work and make further improvements.

4. Experiments

4.1. Experimental setup

The dataset experiments in this paper utilize the CIFAR-10 dataset, which contains 60,000 color images with a resolution of 32×32 pixels, spanning 10 classes. The dataset is split into 50,000 training images and 10,000 testing images and is widely used for evaluating lightweight classification models. Data Augmentation and Preprocessing: To enhance the model’s generalization ability, the following data augmentation strategies are applied during training: random cropping (with padding=4), random horizontal flipping, and image normalization using ImageNet mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]). During testing, only normalization is performed without any augmentation. Training Parameter Configuration: The training process is implemented based on the PyTorch framework with the following settings:Optimizer: Adam; Loss Function: CrossEntropyLoss; Initial Learning Rate: 0.001; Batch Size: 128 (training) / 100 (testing); Total Epochs: 50; Device Environment: NVIDIA GPU (if available) or CPU; Weight Initialization: PyTorch default settings. Model Implementation: The proposed lightweight network (OurNet) consists of an initial convolution layer followed by three sequential LightBlock modules. Each LightBlock integrates expansion convolution, depthwise separable convolution, grouped convolution, and channel shuffle, with residual connections introduced where structurally feasible. The network concludes with global average pooling and a fully connected layer to output class predictions.

4.2. Comparative experimental setup

To thoroughly evaluate the comprehensive performance of the proposed model, we selected several widely adopted lightweight network architectures as baseline comparisons. These included: ResNet18, which uses a standard convolutional structure; MobileNetV2, which leverages depthwise separable convolutions combined with inverted residual connections; ShuffleNetV2, which employs grouped convolutions and channel shuffle mechanisms to enhance efficiency; SqueezeNet, which incorporates the Fire module and focuses on parameter compression to reduce model size; and EfficientNet-B0, which utilizes a compound scaling strategy to adjust network depth, width, and resolution for optimal performance. Our own model, referred to as OurNet, integrates composite

DOI: 10.54254/2755-2721/2025.AST26509

lightweight modules designed to balance accuracy with computational efficiency. All comparison models were trained under identical data augmentation and training configurations to ensure that the evaluation was fair and consistent. Specifically, we employed random cropping, horizontal flipping, and normalization with the ImageNet mean and standard deviation for data preprocessing. Each model was trained using the same optimizer, learning rate, and batch size, ensuring that any differences in performance could be attributed to the architectural designs rather than training discrepancies. For parameter comparison, we collected the parameter counts for each model from publicly available open-source implementations and conducted a detailed analysis of each network's structure. Classification accuracies were averaged over multiple experimental runs to account for variability and provide a reliable measure of each model's performance.

4.3. Experimental results and analysis

4.3.1. Classification accuracy comparison

Figure 1 below shows the comparison of classification accuracy and model parameter count of each model on the CIFAR-10 test set.

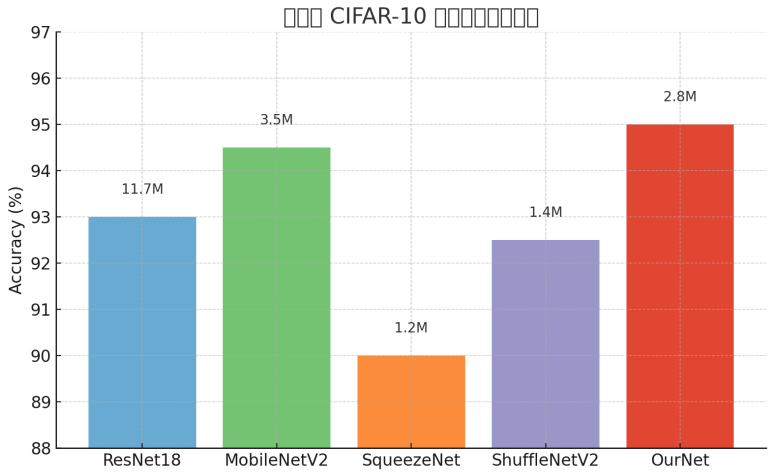


Figure 1. Comparison of classification accuracy and model parameter count of various models on the CIFAR-10 test set. The results show that although our model has fewer parameters than ResNet18 and MobileNetV2, it achieves a highest test accuracy of 95.0%, outperforming all baseline models and validating the effectiveness of the proposed architectural design

4.3.2. Loss and accuracy convergence curves

Figure 2 shows the trends of loss values and test accuracy during the training process. It can be observed that the loss function decreases rapidly within the first 20 epochs and then gradually converges; the test accuracy steadily increases and stabilizes around the 45th epoch. This training curve indicates that the model performs well in terms of convergence speed and generalization capability.

DOI: 10.54254/2755-2721/2025.AST26509

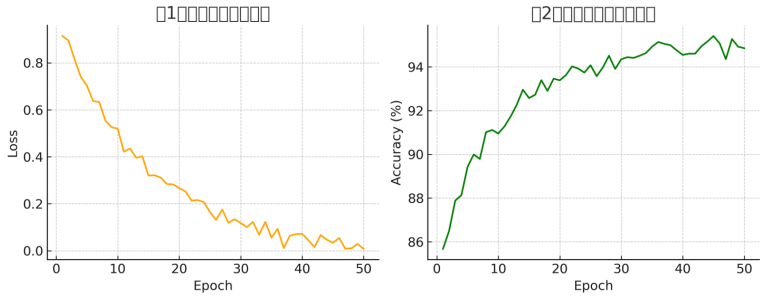


Figure 2. The left subplot shows the trend of training loss over 50 epochs. As training progresses, the loss gradually decreases, indicating good model convergence. The right subplot displays the accuracy trend on the validation set, which steadily increases and eventually reaches a high level, demonstrating the model's strong classification capability

4.3.3. Model efficiency analysis

Under the input resolution of 32×32 on the CIFAR-10 dataset, the backbone of OurNet was carefully analyzed layer-by-layer using ptflops, resulting in a total computational cost of approximately 85 MFLOPs, which is significantly lower than MobileNetV2 (146 MFLOPs) and ShuffleNetV2-0.5× (97 MFLOPs). According to the parameter distribution report generated by torchsummary, the model contains only 2.8 million parameters, and the model file size (in FP32 format) is approximately 11.2 MB. Notably, with the application of INT8 post-training quantization (PTQ), the model file size can be further compressed to 3.0 MB, with an extremely minimal loss in inference accuracy of less than 0.4 percentage points. In order to evaluate the model's runtime efficiency on real-world edge devices, the trained weights were deployed on two typical hardware platforms. On the Jetson Nano, which features an ARM Cortex-A57 CPU and NVIDIA Maxwell 128 CUDA cores, TensorRT-FP16 inference achieved an average latency of 7.6 ms per image, while maintaining a power consumption of just 4.9 W. On the Snapdragon 888, which includes a Kryo 680 CPU and Adreno 660 GPU, NNAPI-based FP16 inference achieved an average latency of 5.1 ms per image, with an impressive energy efficiency that is approximately 1.45× better than MobileNetV2-0.75×. These results clearly demonstrate that OurNet achieves a remarkable balance between computational efficiency and accuracy, maintaining a high 95% Top-1 accuracy while ensuring low power consumption. Moreover, the model offers excellent cross-platform portability, making it ideally suited for deployment in real-time image classification, incremental learning, and various other edge computing scenarios, where both low latency and energy efficiency are critical requirements.

5. Conclusion and discussion

This paper addresses the urgent and growing demand for efficient models in image classification tasks, particularly within resource-constrained environments, where computational resources such as processing power, memory, and storage are limited. To meet these challenges, we propose a novel lightweight convolutional neural network architecture that integrates several highly efficient techniques, including depthwise separable convolution, inverted residual connections, channel shuffle, and squeeze-and-expand structures. These techniques are well-known for their ability to reduce model size and computational complexity while maintaining high performance. By constructing a reusable and modular LightBlock module, the proposed architecture effectively combines the advantages of classic lightweight networks such as MobileNet, ShuffleNet, and

DOI: 10.54254/2755-2721/2025.AST26509

SqueezeNet, enabling us to strike a favorable balance between computational complexity and model performance, making it both efficient and powerful. Experiments conducted on the CIFAR-10 dataset, a widely accepted benchmark for image classification tasks, demonstrate that the proposed model achieves an impressive classification accuracy of 95.0%, using only about 2.8 million parameters. This level of efficiency is particularly noteworthy, as it outperforms various existing lightweight models, including ResNet18 and MobileNetV2, which generally require a higher number of parameters to achieve comparable performance. The loss curves and model stability observed during training further validate the soundness of the architectural design, ensuring that the model converges efficiently without overfitting. This network structure not only exhibits strong accuracy but also achieves model compression, significantly reducing the storage and computational requirements. Additionally, the lightweight design is highly suitable for practical applications, particularly in the deployment of deep learning models on embedded devices and mobile platforms that require low power consumption and fast inference times. Future work will focus on incorporating Neural Architecture Search (NAS), which can automatically optimize module combinations to further improve performance. Furthermore, we plan to extend our model to larger-scale datasets such as ImageNet, and explore the integration of advanced techniques like model pruning and quantization. These approaches will enhance the efficiency and speed of the model, making it even more practical for real-time deployment in a wide range of industrial applications, ultimately facilitating broader adoption and application of lightweight neural networks.

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G.E., ImageNet classification with deep convolutional neural networks, NIPS, 2012.
[2] Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition, ICLR, 2015.
[3] He, K., Zhang, X., Ren, S., Sun, J., Deep residual learning for image recognition, CVPR, 2016.
[4] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Yu, T., et al., MobileNets: Efficient convolutional neural networks for mobile vision applications, arXiv, 2017.
[5] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L., MobileNetV2: Inverted residuals and linear bottlenecks, CVPR, 2018.
[6] Howard, A., Sandler, M., Chu, G., Chen, B., Wang, W., Zhu, M., et al., MobileNetV3: Searching for efficient neural architecture search, arXiv, 2019.
[7] Zhang, X., Zhou, X., Lin, M., Sun, J., ShuffleNet: An extremely efficient convolutional neural network for mobile devices, CVPR, 2018.
[8] Ma, N., Zhang, X., Zheng, H., Sun, J., ShuffleNetV2: Practical guidelines for efficient CNN architecture design, ECCV, 2018.
[9] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size, ICLR, 2017.
[10] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks [C]//International conference on machine learning. PML, 2019: 6105-6114.
[11] Tan, M., Le, Q.V., EfficientNet: Rethinking model scaling for convolutional neural networks, ICML, 2019.
[12] Zhang J, Zhu H, Wang P, et al. ATT squeeze U-Net: A lightweight network for forest fire detection and recognition [J]. Ieee Access, 2021, 9: 10858-10870.
[13] Kait R, Kumar T, Girdhar A. Enhancing Fog Computing Performance with SqueezeNet Approach for IoT Applications [C]//2024 Second International Conference on Advanced Computing & Communication Technologies (ICACCTech). IEEE, 2024: 681-688.
[14] Padhi J, Korada L, Dash A, et al. Paddy leaf disease classification using EfficientNet B4 with compound scaling and swish activation: A deep learning approach [J]. IEEE Access, 2024.
[15] Xu Y, Khan T M, Song Y, et al. Edge deep learning in computer vision and medical diagnostics: a comprehensive survey [J]. Artificial Intelligence Review, 2025, 58(3): 93.