IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 August 2023, accepted 10 October 2023, date of publication 16 October 2023, date of current version 26 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324713

RESEARCH ARTICLE

LA-ShuffleNet: A Strong Convolutional Neural Network for Edge Computing Devices

HUI ZHANG[1], (Member, IEEE), XIAOYANG ZHU[1], BO LI[2], ZHEN GUAN[1], AND WEIMIN CHE[1]

[1]School of Mechanical Engineering, Jiangsu University of Science and Technology, Zhenjiang 212100, China
[2]School of Industrial Software, Henan University of Engineering, Zhengzhou 451191, China

Corresponding author: Hui Zhang (zimmerman@just.edu.cn)

This work was supported in part by the University-Industry Cooperation Research Project in Jiangsu under Grant BY20221435.

ABSTRACT ShuffleNetV2 is a prominent player in the field of lightweight networks and has significant implications for the development of lightweight networks and edge computing. However, it has limitations, as its accuracy falls short compared to other larger models, and it is not friendly to datasets. It loses its advantage for small-sized images with fewer channels. In this study, we analysed the design structure of ShuffleNetV2 and found room for improvement in its computational complexity and accuracy. To further improve its performance, we first upgraded the ShuffleNetV2 network structure based on the lightweight network design criteria and constructed a new network model. Second, we introduced a novel attention module named the Adaptive Pooling Attention Module (APAM) and integrated it with the new network model, constructing a high-performance model referred to as LA-ShuffleNet. Then, we proposed a convolution operation acceleration strategy called Pack. Finally, we combined the two and conducted corresponding tests on the Windows and JETSON platforms. Extensive experiments indicate that our proposed model not only exhibits substantial improvements over the baseline model but also achieves noteworthy enhancements on the ImageNet dataset, with a rise of 1.4% in Top-1 accuracy and 3.6% in Top-5 accuracy, coupled with a reduction of 0.7M parameters. Moreover, its performance surpasses that of certain prevalent lightweight networks, such as MobileNet.

INDEX TERMS Lightweight network, attention module, edge computing, ShuffleNet.

I. INTRODUCTION

After AlexNet's [1] success in the 2012 ImageNet competition, deep neural networks caught researchers' attention and sparked a wave of interest in deep learning. This wave led to the creation of excellent network models such as VGG, GoogleNet, ResNet and others [2], [3], [4], [5], [31]. At the same time, CNNs have continued to expand their visual application scenarios, which include tasks such as image classification, object detection, and natural language processing. However, to further improve accuracy, researchers have chosen to expand the depth and channel quantity, leading to enormous computational costs. Edge computing is a distributed computing architecture that places computing resources and storage resources on edge nodes close to the data source, greatly reducing the delay and network congestion caused by data transmission, improving data security and privacy protection, and supporting more application scenarios, such as the Internet of Things, smart cities, and autonomous driving. Currently, an increasing number of real-time applications based on CNNs are deployed on edge devices with limited computing power, which poses new challenges for the running efficiency and actual cost of the convolutional models. Therefore, research on lightweight models has become increasingly important.

Currently, there are two mainstream approaches to lightweight models, structural lightweight [6], [7], [8], [9], [10], [11], [12], [30], [32] and model compression [13], [14], [15], [16], [17], [18], [19], [20], [25], [26], [27], [28], [29]. The former emphasizes constructing parameter-efficient and high-performing network models using efficient convolution operations and related components, such as MobileNet V1-V3 and ShuffleNet V1-V2 [7], [8], [9], [10], [11], starting from the network structure itself. The latter

The associate editor coordinating the review of this manuscript and approving it for publication was Mingbo Zhao.

dramatically reduces computational costs through knowledge distillation [16], [17], [29], network pruning [13], [14], [15], and other methods without significantly affecting the model performance. Although the two differ in implementation, they complement each other, and their effective combination can result in unexpected benefits.

The attention module is a commonly used module in deep learning. It was originally proposed to solve the translation problem in natural language processing, but now it has been widely used in image processing, speech recognition and other fields. By introducing the attention mechanism, the neural network can selectively focus on important information, so as to achieve better results. Combining it with the network can achieve better performance. Currently common attention modules include self-attention, multi-head attention [33], etc.

To address the low accuracy and unfriendliness to datasets issues in ShuffleNet V2, this study proposes, by optimizing its structure and making it combine with attention module, an improved model, called LA-ShuffleNet. In the process of enhancing the structure, we combined lightweight design with model compression techniques by utilizing a pruning strategy to remove modules that negatively impact the overall performance of the model. In order to enhance the model's performance and effectiveness, we introduce an attention module termed Adaptive Pooling Attention Module (APAM). To tackle the issue of reduced model expressiveness resulting from pruning, we enlarge the convolutional kernel. Moreover, to tackle the issue of long running times or even the inability to run on edge computing devices during training and testing, we propose a convolution acceleration strategy, named Pack.

II. RELATED WORK

A. LIGHTWEIGHT NETWORK DESIGN

To implement a lightweight design of the network structure, there are various possible implementations, such as using low-dimensional filters (such as 1×1) to reduce the weight parameters, which has been widely used in SqueezeNet [6] and greatly reduces the number of parameters. Group convolution is an important method in current lightweight network design [10], [11]. ShuffleNet V1 [10] uses group convolution instead of 1×1 convolution and proposes channel shuffling to improve the information interaction between channels, while ShuffleNet V2 [11] introduces channel splitting and adjusts the position of channel shuffling based on ShuffleNet V1, achieving a better performance. Deep convolution computes feature values instead of raw values, reducing the computational costs. MobileNet V1 [7] designs a model suitable for mobile devices using deep convolutions. ResNet [4] and MobileNet V2 [8] solve the problem of increased computational cost caused by channel expansion through bottleneck modules. Xception [12] improves deep convolutions by performing pointwise convolutions first. MobileNet V3 [9] introduces NAS (network architecture search) and SE (squeeze-and-excitation) attention mechanisms, that demonstrate excellent performance and speed.

B. MODEL COMPRESSION

Model compression is another mainstream approach in lightweight network design, that plays an important role in reducing the size of pretrained models. Pruning [13], [14], [15] reduces the model size and computational cost by removing redundant channels and connections. Knowledge distillation [16], [17] involves training a large neural network and a small neural network together, transferring knowledge from the ''teacher network'' to the ''student network'' to reduce the generation of redundant information. Model quantization [18], [19] converts floating-point calculations into low-bit fixed-point calculations, effectively reducing parameter quantity and memory consumption. Attention Transfer [20] significantly improves the performance of the ''student network'' by imitating the attention map of the powerful ''teacher network''.

C. ATTENTION MODULE

As a widely adopted technique in current research, the attention module is based on the fundamental concept of dividing input data into two segments: one consisting of essential information that requires attention, and the other comprising non-essential information that does not warrant attention. Transformer [33], as a novel neural network model, has achieved remarkable success in various tasks by replacing traditional convolutional and recurrent structures with self-attention mechanism. The Non-local Neural Networks model [34] employs non-local operations to enable interactions among global features. Meanwhile, the Convolutional Block Attention Module (CBAM) [35] utilizes an attention module consisting of two branches, channel attention and spatial attention, to capture both global and local information. The Squeeze-and-Excitation Networks (SENet) [36] model introduces squeeze and excitation operations to learn the importance of each channel.

D. ACCELERATION OF CONVOLUTION OPERATIONS

The traditional convolution calculation involves sliding a window over the input feature map and performing dot products, which consumes a significant amount of time and resources on data addressing and reading operations. This phenomenon facilitates the emergence of fast convolutional networks. Low-rank decomposition [21] specifies the convolution kernel matrix by merging dimensions and imposing low-rank constraints, resulting in a reduction in storage space. FFT [22] uses related convolution kernels to quickly compute linear convolutions of finite-length sequences, but its computational complexity is high and it increases the memory bandwidth requirements, so it is not widely used. Winograd [23] and FFT [22] are both linear transformations, but the former transforms to a real field without complex multiplication, making it more suitable for small-kernel convolution.

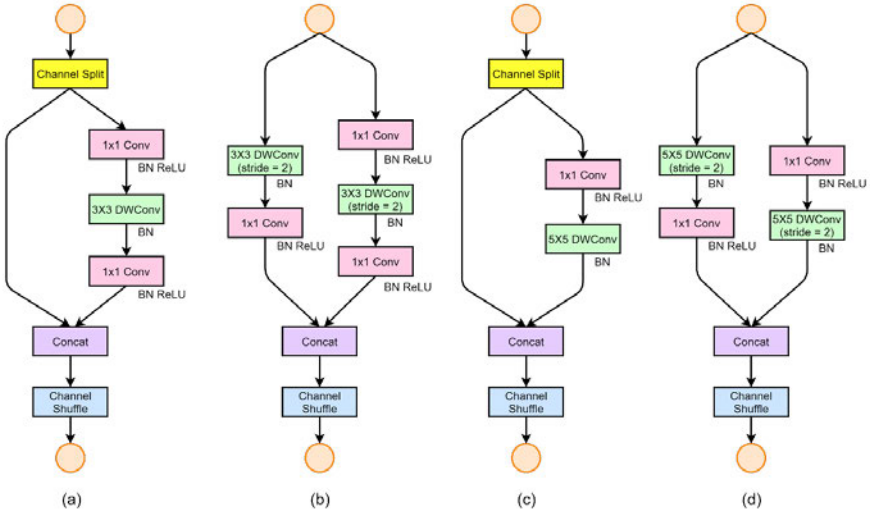


FIGURE 1. Original network structures and our proposed network structure.

TABLE 1. Resource per layer in MobileNets.

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

The prevalent approach typically involves utilizing Im2Col in conjunction with GEMM [24], which entails leveraging Im2Col to reorganize the parameter positions required for convolution calculations, while concurrently exploiting the multithreaded operations supported by GEMM. The combination effectively improves the cache utilization and convolution speed.

III. METHOD

A. IMPROVEMENT OF NETWORK STRUCTURE

To reduce the parameters and computational cost of the model, ShuffleNetV2 proposes two block structures, as shown in (a) and (b) of figure 1. The structure in (a), called the downsampling block, is used when DWConv stride=1 and requires channel splitting and dividing the input feature into two parts. The structure in (b), called the regular block, is used when DWConv stride=2 and does not require channel splitting, instead, it performs direct computations. Following the completion of computation for both branches, regardless of whether it was a downsampling block or a regular block, subsequent concatenation and channel shuffling operations are needed.

Assuming that the left branch is Branch1 and the right branch is Branch2, both the downsampling block and regular block in Branch2 of ShuffleNetV2 use DepthWise convolutions with 1×1 convolutions before and after. Typically, 1×1 convolutions are used before and after a DepthWise convolution for two purposes, first, to merge the information between channels and compensate for the lack of information fusion between the channels in the DepthWise convolution; second, to increase or decrease dimensionality, such as in the inverted residual module in MobileNet V2 [8]. However, using multiple 1×1 convolution kernels significantly increases the computational complexity. Table 1 in the MobileNets [7] paper shows the percentage of multiplication-addition operations and the number of parameters for each layer. Due to the heavy use of 1×1 convolution kernels, most of the computation and parameter quantity in the entire network is concentrated on the 1×1 convolution module, affecting the network performance. In ShuffleNetV2, 1×1 convolutions are used before and after the DepthWise convolution in the right branch, which not only imposes a heavier computational burden, but also diminishes the interpretability of the model. Therefore, we remove the 1×1 convolution after the DepthWise convolution, as shown in the right branch of (c) and (d) in the figure1.

Based on the data in Table 1 and the computation distribution of ShuffleNetV2, we can see that most of the

computation is concentrated on 1 × 1 convolutions, with a small proportion of the computations allocated to DepthWise convolutions. Therefore, we can expand the 3 × 3 convolution kernel into a 5 × 5 convolution kernel, which will not significantly increase the computational complexity while improving network efficiency. The specific structure is shown in (c) and (d) of Figure 1, and we will explore this further in subsequent experiments.

B. NEW ATTENTION MODULE

The attention module focuses attention on important parts or features of the input data to enhance the performance and effectiveness of the model. Based on the aforementioned roles, we have constructed a novel attention module called the Adaptive Pooling Attention Module (APAM). The structure of this module, as shown in figure 2(a), includes two pooling operations (average pooling and max pooling) and two convolutional layers. The input feature map is subjected to average pooling and max pooling to obtain its maximum and average values along the channel dimension, respectively. These two results are then fused through two convolutional layers, and finally passed through a sigmoid activation function to generate weights, which are used to weight the original feature map and produce the attention feature map. Next, we combine it with the new block mentioned in Section III-A, as shown in (b) of figure 2, which involves residual connections. After performing the Channel Split operation, it is divided into two branches. The left branch does not undergo any operations, while the right branch connects the attention module to the original structure through a residual module. The output of the original structure is then combined with the output of the attention module. This process results in a novel block that incorporates residual connections and attention modules.

C. IMPROVEMENT IN THE CONVOLUTIONAL OPERATION

The Im2Col method converts convolution operations into a matrix multiplication. From the perspective of the number of addition and multiplication operations, there is no difference compared to traditional convolution operations. However, after conversion into a matrix, data are stored in contiguous memory, resulting in improved access speeds. For large convolution kernel sizes and output feature map channels, the Im2Col method often generates matrices with more rows than columns. Due to the data redundancy phenomenon in the computational process, there is redundant information in row storage during data reading, which affects the calculation speed, as shown in (a) and (b) of figure 3.

This study utilizes a method called data packing (Pack) to address the data redundancy issue in Im2Col. This method compresses the row dimension and expands the column dimension, as shown in figure 3. In figure 4(a), a 3 × 3 convolution kernel with 5 output channels is used as an example to illustrate the matrix form obtained through the

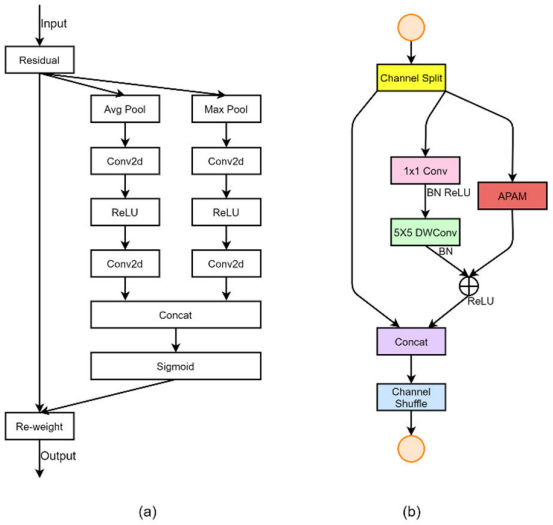
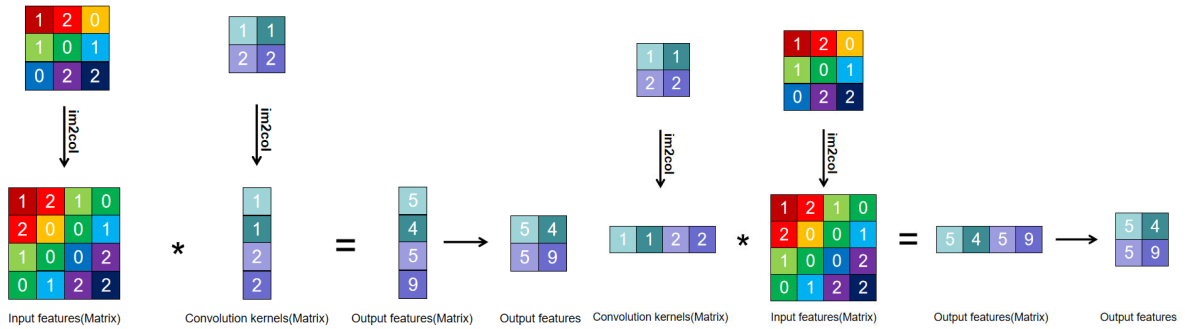


FIGURE 2. Proposed APAM attention module.

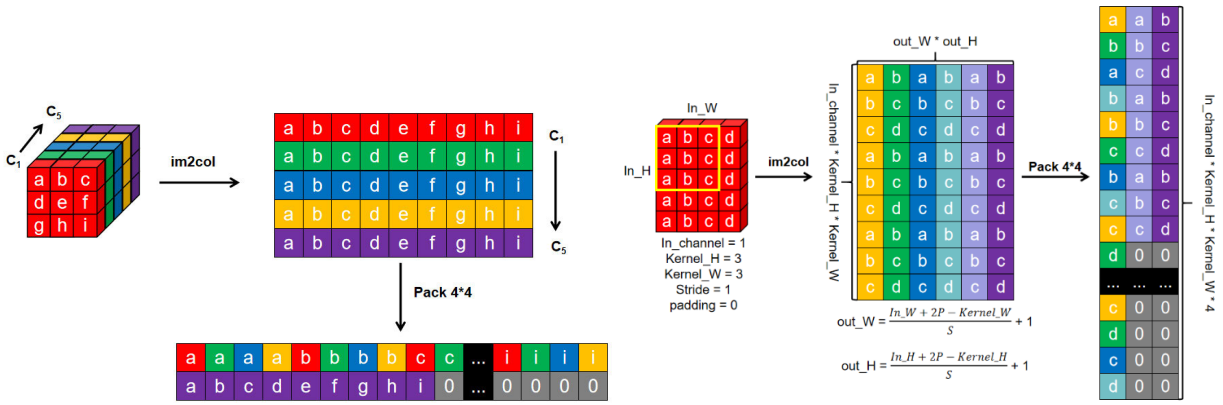
Im2Col method. Since the framework used in this study unfolds the sliding windows into rows, the colours of each channel correspond to the expanded rows. The 4×4 operation is used in figure 4(a). Since the output channel number cannot be divided by 8, the output channel number is 5, which is divided by 4 with a remainder of 1. Therefore, elements are horizontally expanded in groups of four in the column direction, with red, green, blue, and yellow forming one group. After integrating the first four rows of elements, the purple elements in the fifth row are placed in the second row of the output result, and zero padding is applied where needed to fill in missing elements. If there is a remainder, additional row data are supplemented below horizontally, and zero padding is applied to fill in any missing elements. If it can be evenly divided, the output is a one-dimensional vector with multiple columns, effectively reducing computational complexity.

Figures 4(b) and 4(c) illustrate the 4 × 4 and 8 × 8 Pack operations applied to the input feature vectors. In the sequel, we shall employ the Jetson platform to validate the soundness of our proposed concept. Similar to the Pack operation for convolution kernels, the Im2Col method is used to convert input features into a matrix form. Since the convolution kernel is unfolded row by row, the input feature needs to be expanded column by column, as shown in figure 4(b). Then, the Pack operation is performed. Taking figure 4(b) as an example, the horizontal dimension of the feature matrix is divided by 4 with a remainder of 2, so elements are horizontally expanded in groups of four in the column direction, with each group forming one column after repeated operations. The last two columns of the input feature matrix are placed in the second and third columns of the output, respectively, and missing elements are filled with zeros. The basic process for

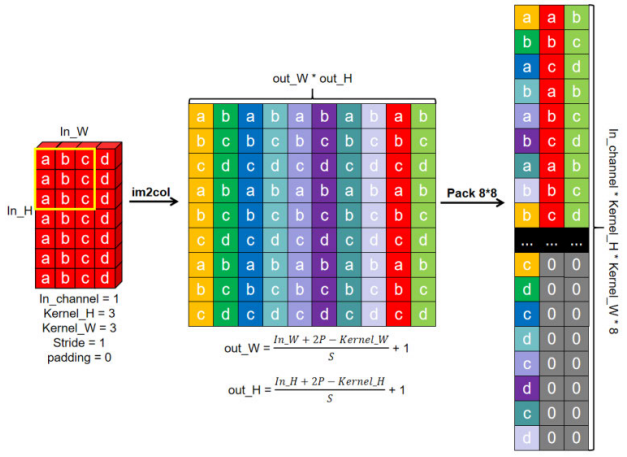


(a) Convolution kernel expanded by column (b) Convolution kernel expanded by row

FIGURE 3. Two mainstream methods of Im2Col.



(a) Examples of Pack methods (b) Operation of Pack 4×4



(c) Operation of Pack 8×8

FIGURE 4. Packaging strategy implementation.

figures 4(b) and 4(c) is the same, except that they use 4 or 8 as the base.

D. EFFICIENCY ANALYSIS

Section III-A mainly focuses on the improvements made to the Branch2 block structure in ShuffleNetV2, and the following analysis is based on the computational complexity. The specific process involved in Branch2 of the original ShuffleNetV2 structure can be illustrated by the diagram below, assuming that M feature maps of size Dp x Dp are used as input. The specific process is shown in figure 5:

The total calculation amount brought by the above three stages is:

$$D_p^2 * M * N + D_f^2 * D_k^2 * N + D_f^2 * N * K \qquad (1)$$

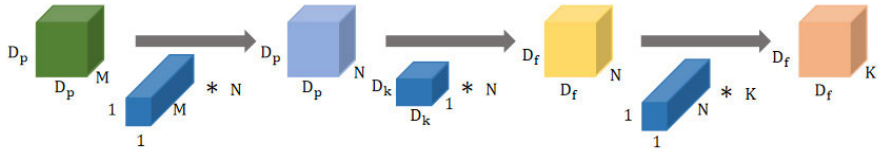


FIGURE 5. Specific process of Branch2 in ShuffleNetV2.

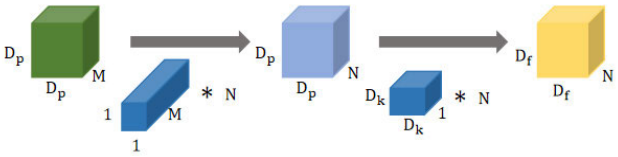


FIGURE 6. Specific process of Branch2 in LA-ShuffleNet.

The specific process involved in the improved Branch2 is shown in the following figure 6. The total calculation amount brought by the above two stages is:

$$D_p^2 * M * N + D_f^2 * D_k^2 * N \tag{2}$$

To ensure consistency in output size, it is recommended to set the kernel size K to be equal to the input size N, and divide both Formula (1) and Formula (2) by this value.:

$$\frac{D_p^2 * M * N + D_f^2 * D_k^2 * N + D_f^2 * N^2}{D_p^2 * M * N + D_f^2 * D_k^2 * N} \tag{3}$$

Under the condition that the size of the convolution kernel remains unchanged, the simplified formula of Formula (3) is:

$$1 + \frac{D_f^2 * N^2}{D_p^2 * M * N + D_f^2 * D_k^2 * N} \tag{4}$$

It is obvious that Formula (4) is always greater than 1, that is, deleting the 1×1 convolution operation can reduce the calculation amounts.

In the original network structure, the size of the convolution kernel is 3×3, which can be substituted into Formula (1), and D_k is 3, while in our improved network, its size is expanded to 5×5, and can be substituted into Formula (2), and D_k is 5. Then, Formula (3) can be simplified as follows:

$$1 + \frac{(N - 16)\, D_f^2}{D_p^2 * M + 25 * D_f^2} \tag{5}$$

Table 2 displays the network structure of ShuffleNetV2, and we can observe that the value of N in the "Output channels" column is always greater than 16, indicating that the value in Equation (5) is always greater than 1. This indicates that removing the redundant 1×1 convolution layers and enlarging the DepthWise convolution kernels can greatly reduce the computational complexity. Therefore, our proposed improvements can theoretically produce a positive effect. In the next step, we will conduct relevant experiments to further investigate this.

IV. EXPERIMENT AND RESULTS

A. OPERATING ENVIRONMENT AND DATASETS

1) OPERATING ENVIRONMENT

In terms of the experimental devices utilized, we provide detailed information for the Windows platform, as illustrated in table 3.

The detailed configuration for the JESTON platform is presented in table 4.

2) DATASETS

We leverage the ImageNet dataset to rigorously assess the performance of our proposed model.

The ImageNet dataset [38], a pioneering benchmark in the field of computer vision, has played a pivotal role in advancing the development and evaluation of various image recognition algorithms. Comprising over millions of labeled images across thousands of categories, ImageNet provides an extensive and diverse collection of visual data. Its widespread adoption has enabled researchers to explore the frontiers of deep learning, fostering innovation in object detection, image classification, and semantic segmentation. The challenge of achieving high accuracy on the ImageNet dataset has driven the design of intricate neural architectures, such as convolutional neural networks (CNNs) and their variants, contributing to significant breakthroughs in the field. As illustrated in Figure 7, due to its extensive composition of 1000 major categories, a subset of 10 classes has been chosen for demonstration purposes in this instance.

B. EVALUATION OF THE CONVOLUTION OPTIMIZATION METHODS

Regarding the optimization of the convolution method in Section III-B, we conducted the following tests, and the results are shown in table 5. We conducted a comparative analysis of our proposed method against manual optimization as well as mainstream Winograd acceleration methods. The input size for the first three sets of data is 15×15, with 512 input channels, 1024 output channels, and a 3×3 kernel size. It can be seen from the time used that our method achieved better results. For the last three sets of data, we increased the input size while reducing the input and output channels and conducted tests accordingly. Our method also achieved good results.

C. EVALUATION OF THE ATTENTION MODULE

In Section III-B, we introduce a novel attention module named Adaptive Pooling Attention Module (APAM).

TABLE 2. Network structure parameters of ShuffleNetV2.

Layer	Output size	KSize	Stride	Repeat	Output channels			
					0.5×	1.0×	1.5×	2.0×
Image	224×224				3	3	3	3
Conv1 MaxPool	112×112 56×56	3×3 3×3	2 2	1	24	24	24	24
Stage2	28×28 28×28		2 1	1 3	48	116	176	244
Stage3	14×14 14×14		2 1	1 7	96	232	352	488
Stage4	7×7 7×7		2 1	1 3	192	464	704	976
Conv5	7×7	1×1	1	1	1024	1024	1024	1024
GlobalPool	1×1	7×7						
FC					1000	1000	1000	1000

TABLE 3. The configuration under the windows platform.

Hardware		Software	
CPU	Intel(R) Core(TM)i7-9700CPU	OS	Windows10
RAM	32GB	CUDA Toolkit V11.1 CUDNN 8.0.4 Python 3.7 PyTorch 1.8.0	
GPU	NVIDIA GeForce RTX 3060		

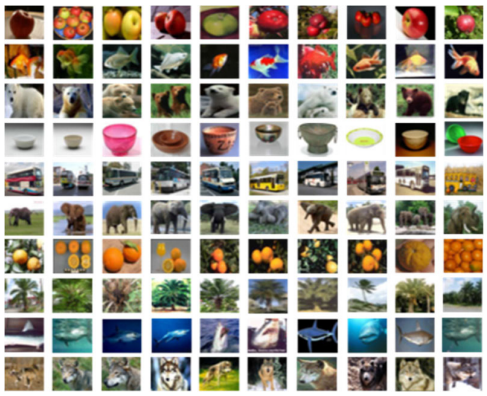


FIGURE 7. Examples of images in the dataset.

We conducted evaluations on the ImageNet dataset and employed Grad-CAM for visual representation, with specific results presented in Figure 8. Our experimentation involves ten distinct image categories as test subjects, with the Grad-CAM [37] tool employed to showcase the network's focus. Notably, when examining the original ShuffleNetV2 network, it becomes evident that certain crucial features, such as the Binder, Jesery, and Vase, are not adequately attended to, as indicated by significant discrepancies in these images. Conversely, our model, integrated with the APAM module, exhibits improved capability to concentrate on important features. For instance, in the case of the Binder category, our model effectively focuses on the intrinsic characteristics of the Binder itself while acquiring a broader range of features and paying increased attention to significant aspects.

D. EVALUATION OF NETWORK PERFORMANCE

1) TRAINING SETTINGS

According to our repeated experiments, the best configuration for training is as follows:

Our model was trained for 300 epochs on the ImageNet dataset using the stochastic gradient descent (SGD) optimizer with a batch size of 128. We utilized cosine learning rate decay with an initial learning rate of 0.01 and a momentum parameter of 0.9 (Nesterov set to true). Weight decay was applied to the model with a value of 1e-4 to improve performance.

2) TEST UNDER WINDOWS PLATFORM

In table 6, we present the experimental results on the Windows 10 platform, encompassing MACs, Parameters, Top-1 accuracy, and Top-5 accuracy. In comparison with the classical convolutional neural networks, namely AlexNet and VGG, the lightweight networks exhibit superior performance under relatively lower computational and parameter constraints. Notably, when compared with the ShuffleNet series, enhancements in accuracy have been achieved. Furthermore, in contrast to the MobileNet series, improved performance is attained with reduced computational and parameter demands. The performance of LA-ShuffleNet surpasses that of SqueezeNet and approaches the level of DenseNet with lower computational requirements, yet demonstrates a disparity when contrasted with EfficientNet.

3) TEST UNDER THE JETSON PLATFORM

In table 7, we present the experimental results on the JETSON XAVIER NX development kit environment, encompassing MACs, Parameters, Top-1 accuracy, and Top-5 accuracy. In comparison with the experimental outcomes on the Windows 10 platform, our model continues to demonstrate outstanding performance.

TABLE 4. The configuration under the JESTON platform.

Hardware		Software	
CPU	NVIDIA Carmel ARMv8.2 (6-core)@1.4GHz(6MB L2 + 4MB L3)	OS	Ubuntu 18.04
RAM	8GB 128-bit LPDDR4x @1600 MHz|51.2GB/ s	CUDA Toolkit V10.2 CUDNN 8.0 Python 3.7 PyTorch 1.8.0	
GPU	384-core Volta @1100 MHz + 48 Tensor Cores		

TABLE 5. The time comparison of different convolution methods.

Shape	InputChannel	KSize	OutChannel	Optimization Algorithm	Time
15×15	512	3×3	1024	manual optimization	578.47ms
15×15	512	3×3	1024	WinoGrad	334.85ms
15×15	512	3×3	1024	Im2col+Pack+Sgemm	302.45ms
56×56	64	3×3	128	manual optimization	122.64ms
56×56	64	3×3	128	WinoGrad	61.33ms
56×56	64	3×3	128	Im2col+Pack+Sgemm	57.51ms

TABLE 6. Comparison of model performances on ImageNet under Windows 10 platform.

Model	MACs(M)	Parameters(M)	Top-1 Accuracy(%)	Top-5 Accuracy(%)
LA-ShuffleNet	**157**	**3**	**71.9%**	**91.7%**
AlexNet	710	6	56.9%	79.2%
VGG19	1960	143	71.5%	90.3%
ShuffleNetV1	140	5.1	67.1%	86.9%
ShuffleNetV2	140	3.7	69.5%	88.1%
MobileNetV1	320	4.3	70.1%	88.9%
MobileNetV2	350	3.5	71.5%	90.2%
MobileNetV3 Small	300	2.3	66.9%	79.1%
MobileNetV3 Large	520	5.5	73.5%	90.5%
SqueezeNet	83.7	1.4	56.8%	79.8%
DenseNet	780	79.7	74.3%	91.6%
EfficientNet	39	5.2	76.1%	92.8%

TABLE 7. Comparison of model performances on ImageNet under Jeston platform.

Model	MACs(M)	Parameters(M)	Top-1 Accuracy(%)	Top-5 Accuracy(%)
LA-ShuffleNet	**157**	**3**	**71.5%**	**91.2%**
AlexNet	710	6	56.8%	78.5%
VGG19	1960	143	71.5%	90%
ShuffleNetV1	140	5.1	66.9%	85.7%
ShuffleNetV2	140	3.7	69.1%	87.6%
MobileNetV1	320	4.3	70%	88.8%
MobileNetV2	350	3.5	71.3%	90.1%
MobileNetV3 Small	300	2.3	66.9%	78.7%
MobileNetV3 Large	520	5.5	73.2%	90.1%
SqueezeNet	83.7	1.4	56.5%	79.5%
DenseNet	780	79.7	73.9%	91%
EfficientNet	39	5.2	76%	92.3%

E. ABLATION STUDY

The incorporation of lightweight and attention modules holds significant importance in the architecture of LA-ShuffleNet, as they directly contribute to the overall performance and efficiency of the network. To gain deeper insights into the influence of these modules, we conducted a comparative analysis, examining their performance across three distinct network bandwidths: 0.5x, 1.0x, and 1.5x, both with and without the presence of these two conditions. Table 8 presents the pertinent models deployed in the assessment.

The test results of the four models mentioned above on ImageNet are shown in figure 9. After making the network lighter, the Top-1 accuracy is lower than that of the original network. With the introduction of attention mechanism, the accuracy has been improved to some extent, but it is still not

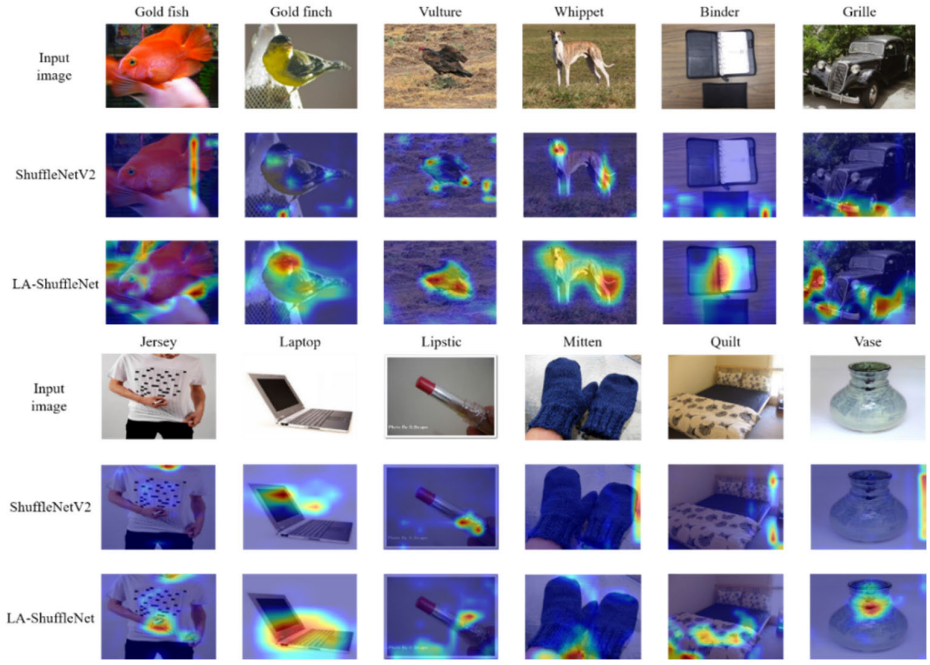


FIGURE 8. Heat map generated by Grad-CAM tool.

as good as L4, the LA-ShuffleNet designed by combining lightweight and attention mechanism.

In conclusion, only by combining lightweight design with attention module, the network can achieve optimal performance. Using a single module alone cannot achieve the desired effect. Additionally, our proposed LA-ShuffleNet has achieved excellent performance across different network widths.

V. DISCUSSION

Based on the results reported in the "EVALUATION OF NETWORK PERFORMANCE" section and the relevant data from Tables 6 and 7, our study strongly exhibits the remarkable reliability and stability of the proposed LA-ShuffleNet. This performance makes it a suitable alternative for deployment on multiple terminal devices running different operating systems. The experimental findings indisputably suggest that the proposed network model outperforms the standard and well-established AlexNet under identical testing settings. This finding not only verifies the superiority of our model but also emphasizes its potential uses in edge computing situations.

It is worth noting that LA-ShuffleNet maintains its outstanding performance when compared to VGG19, even while considerably reducing computational costs. Furthermore, it outperforms current mainstream lightweight networks, such as the ShuffleNet series and the MobileNet series. However, it is vital to recognize that some performance gap still persists when compared to EfficientNet. This provides valuable directions for future research and improvement.

TABLE 8. Four models used for ablation experiments.

Model	Lighter design	Attention module
L1	NO	NO
L2	YES	NO
L3	NO	YES
L4	YES	YES

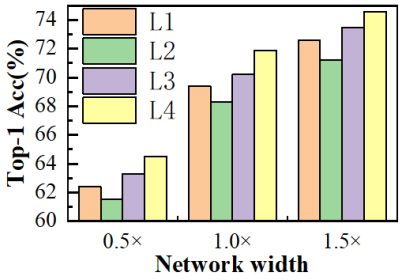


FIGURE 9. Performance of four models on ImageNet.

Our study's conclusions are further strengthened by the findings in the ablation study section. The model LA-ShuffleNet exhibits exceptional performance on both the Windows 10 platform and the JETSON XAVIER NX development kit when utilized with convolutional optimization methods and attention modules. This placed LA-ShuffleNet at the forefront of edge computing devices, creating a solid foundation for practical implementation.

Nevertheless, despite comparable results on both platforms, it is worth noting that there is a significant difference in actual training time. Specifically, training on the

JETSON platform takes 1.5 to 2 times longer than on the Windows 10 platform. This discrepancy is mostly related to the restricted computational capabilities of mobile devices and the associated increase in overall runtime. This difference provides useful information for resource planning and time budgeting in actual applications to ensure optimal performance and efficiency on different platforms. Future study should further examine techniques to optimize training on mobile devices, minimize training time, and enhance computational efficiency to satisfy the expanding demands of edge computing.

VI. CONCLUSION

In this study, we introduce LA-ShuffleNet, a more efficient model built on ShuffleNetV2 with an enhanced internal structure. Our results reveal that LA-ShuffleNet outperforms the ShuffleNet series and other popular lightweight networks in terms of both accuracy and efficiency. Moreover, LA-ShuffleNet is compatible with numerous platforms, including mobile devices. Our findings offer insights for future lightweight model designs.

REFERENCES

[1] A. Krizhevsky, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] K. M. He, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[9] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[10] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[11] N. N. Ma, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 116–131.

[12] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016, *arXiv:1610.02357*.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[14] S. Han, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[15] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," 2017, *arXiv:1708.06519*.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[17] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," 2020, *arXiv:2006.05525*.

[18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017, *arXiv:1712.05877*.

[19] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4820–4828.

[20] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," 2016, *arXiv:1612.03928*.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.

[22] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," 2013, *arXiv:1312.5851*.

[23] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4013–4021.

[24] M. Cho and D. Brand, "MEC: Memory-efficient convolution for deep neural network," 2017, *arXiv:1706.06873*.

[25] W. Jiang, "Hybrid computational strategy for deep learning based on AVX2," *J. Tsinghua Univ. Sci. Technol.*, vol. 60, no. 5, pp. 408–414, 2020.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[27] I. Fostiropoulos and B. Boehm, "Implicit feature decoupling with depth-wise quantization," 2022, *arXiv:2203.08080*.

[28] Y. Zhong, M. Lin, G. Nan, J. Liu, B. Zhang, Y. Tian, and R. Ji, "IntraQ: Learning synthetic images with intra-class heterogeneity for zero-shot network quantization," 2021, *arXiv:2111.09136*.

[29] L. Zhang, X. Chen, X. Tu, P. Wan, N. Xu, and K. Ma, "Wavelet knowledge distillation: Towards efficient image-to-image translation," 2022, *arXiv:2203.06321*.

[30] L. Ye, "AugShuffleNet: Communicate more, compute less," 2022, *arXiv:2203.06589*.

[31] H. Zhang, R. Luo, L. Luo, K. Li, X. Fang, and S. Zhang, "Deep learning for drawing numbering in engineering drawing management: A case study for refrigerated compartment product," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 37, no. 4, Mar. 2023, Art. no. 2352005.

[32] W. Zhu, H. Zhang, J. Eastwood, X. Qi, J. Jia, and Y. Cao, "Concrete crack detection using lightweight attention feature fusion single shot multibox detector," *Knowl.-Based Syst.*, vol. 261, Feb. 2023, Art. no. 110216.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[34] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.

[35] S. Woo, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 3–19.

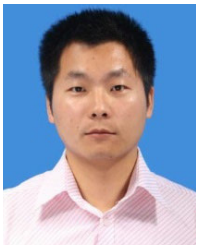
[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," 2016, *arXiv:1610.02391*.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

HUI ZHANG (Member, IEEE) received the Ph.D. degree in aerospace manufacturing from the Nanjing University of Aeronautics and Astronautics, in 2009. He is currently an Associate Professor with the School of Mechanical Engineering, Jiangsu University of Science and Technology. His research interests include intelligent manufacturing, reverse engineering, machine vision, deep learning, fault diagnosis, non-contact measurement, and AR.

XIAOYANG ZHU received the bachelor's degree in software engineering from the Tongda College, Nanjing University of Posts and Telecommunications, in 2021. His current research interests include deep learning and human pose estimation.



BO LI received the Ph.D. degree in aeronautical and astronautical engineering from the Nanjing University of Aeronautics and Astronautics, in 2013. He is currently a Teacher with the Institute of Industrial Software, Henan University of Engineering. His current research interests include industrial intelligence and data management.

ZHEN GUAN received the bachelor's degree in mechanical engineering from the Jiangsu University of Science and Technology, in 2021. His current research interests include intelligent manufacturing and machine vision.

WEIMIN CHE received the bachelor's degree in material forming and control engineering from Tangshan University, in 2021. His current research interests include augmented reality and machine vision.