Pattern Analysis and Applications (2024) 27:123
https://doi.org/10.1007/s10044-024-01327-3

SHORT PAPER

ShuffleNeMt: modern lightweight convolutional neural network architecture

Meng Zhu[1] · Weidong Min[1,2,3] · Qing Han[1,2,3] · Guowei Zhan[1] · Qiyan Fu[1] · Jiahao Li[1]

Received: 9 April 2023 / Accepted: 23 August 2024

Abstract
Lightweight convolutional neural networks (CNNs) are specifically designed for mobile devices and embedded systems, being aimed at achieving lower resource and faster inference speed. However, these networks are limited in their ability to capture long-range dependencies due to the local nature of convolutional operations. The introduction of self-attention into these CNNs can effectively capture global information, but it comes at the cost of significantly slower inference speed. To solve these problems, we propose a novel lightweight network called as ShuffleNeMt. Our ShuffleNeMt involves modernizing ShuffleNetV2 by incorporating several strategies, including pre-norm residual learning, scaling the residual depth, visual self-attention and non-monotonic activation functions. The visual self-attention is used for only one layer and positioned in the middle of the network. Using this way, ShuffleNeMt not only can achieve efficient resource utilization and fast inference speed, but also can capture long-range spatial dependencies. Extensive experimental results demonstrate the superiority of ShuffleNeMt over the existing lightweight architectures. E.g., On the CIFAR-100 image classification dataset, ShuffleNeMt-1.5 achieved top-1 error rate of 34.05% using 2.588M parameters, which is better than the top-1 error rate of 36.86% achieved by MobileNetV3-Large-0.8 using 2.809M parameters.

Keywords Convolutional neural networks · Self-attention · Lightweight

1 Introduction

In the past decade, deep neural networks have revolutionized computer vision, improving abilities in image classification [1, 2], object detection [3, 4] and video analysis [5]. Network structures have evolved quickly, resulting in significant advancements such as AlexNet [1], GoogleNet [6], ResNet [2] and EfficientNet [7]. These progressions have notably boosted the accuracy and efficiency of various visual tasks, elevating their overall performance.

Modern neural networks have exceeded human accuracy in image recognition, achieving over 95% classification accuracy in the ImageNet Image Classification Challenge [2]. They have also made significant advances in real-time machine translation and autonomous driving technology, matching human speed and quality in translation [8], and making high-speed decisions in autonomous driving based on real-time sensor data analysis [9, 10]. When deploying neural networks on edge devices such as mobile devices and embedded systems, it is essential to take into account not just the performance of the model, but also its efficiency, particularly the actual computational resource and inference speed. Developing lightweight convolutional neural networks (CNNs) has been an effective way to reduce computational resource and inference latency [11–20].

However, due to the requirement of computational resource and inference speed, most of these lightweight CNNs cannot be designed too deep. Constrained by the fact that convolutional operations can only capture local information within a window region, most of these lightweight CNNs have limited capability to capture long-range spatial dependencies, which hinders further performance improvement. Combining self-attention operations with convolutional operations can capture global spatial dependencies,

✉ Weidong Min
minweidong@ncu.edu.cn

1 School of Mathematics and Computer Science, Nanchang University, Nanchang 330031, China

2 Institute of Metaverse, Nanchang University, Nanchang 330031, China

3 Jiangxi Provincial Key Laboratory of Virtual Reality, Nanchang University, Nanchang 330031, China

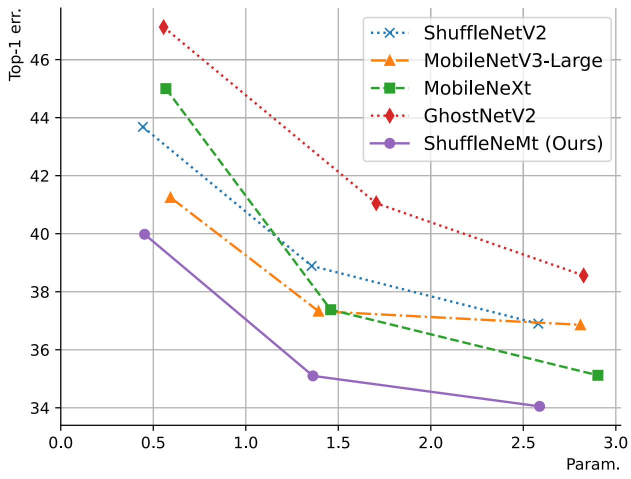


Fig. 1 Trade-off between parameters and top-1 error rate

but self-attention comes with computational complexity of $\mathcal{O}(CH^2W^2)$.

To solve these problems, an intuitive solution is to combine a lightweight CNN model with visual self-attention to enhance the model's ability to capture global spatial dependencies, while avoiding the computational complexity of visual self-attention. Based on this idea, we propose a lightweight CNN model called ShuffleNeMt, whose network architecture is shown in Table 1. In the table, the Local Residual Block and Local ShuffleNeMt Block are responsible for capturing local spatial dependencies through convolutional operators. The Non-local ShuffleNeMt Block is responsible for capturing global spatial dependencies through visual self-attention, and is used only once in the middle of the network. With this approach, ShuffleNeMt not only maintains its lightweight characteristics but also has the capability to capture global spatial dependencies. Figure 1 plots the trade-off between parameters and top-1 error rate on CIFAR-100 [21] dataset. As shown in Fig. 1, it can be inferred that among the five lightweight CNN models compared, ShuffleNeMt is the optimal model given the same parameter complexity.

The contributions of this paper can be summarized as follows.

1. We propose ShuffleNeMt, a novel lightweight architecture that incorporates modern network design principles. ShuffleNeMt not only achieves low computational resource and high inference speed, but also captures long-range spatial dependencies.
2. We conduct comparative experiments on the CIFAR-100 [21] image classification, the PASCAL VOC 2007/2012 [22] object detection and CityScapes [23] semantic segmentation datasets to verify the superiority of our ShuffleNeMt.

The rest of the paper is organised as follows. Section 2 reviews the related work. Section 3 introduces the proposed ShuffleNeMt in detail. Section 4 shows and discusses the experimental results. Section 5 draws the conclusion.

2 Related work

Lightweight CNNs are aimed at achieving the optimal trade-off between computational resources and accuracy, making them be widely-applied into mobile devices and embedded

Table 1 ShuffleNeMt baseline network. Each row describes a stage i with L_i layers, with output resolution $H_i \times W_i$ and output channels C_i

Stage i	Operator F_i	Resolution $H_i \times W_i$	#Channels C_i				#Layers L_i
			0.5×	1.0×	1.5×	2.0×	
1	Stacked SDB	56×56	24	24	24	24	1
2	Local residual block	56×56	24	24	24	24	2
3	Stepwise SDB	28×28	48	116	176	244	1
4	Local ShuffleNeMt block	28×28	48	116	176	244	3
5	Stepwise SDB	14×14	96	232	352	488	1
6	Local ShuffleNeMt block	14×14	96	232	352	488	3
7	Non-local ShuffleNeMt block	14×14	96	232	352	488	1
8	Local ShuffleNeMt block	14×14	96	232	352	488	2
9	Stepwise SDB	7×7	192	464	704	976	1
10	Local ShuffleNeMt block	7×7	192	464	704	976	3
11	BN, nonlinear	7×7	192	464	704	976	1
12	1×1 Conv, nonlinear, dropout	7×7	1024	1024	1024	2048	1
13	GAP, FC	1×1	k	k	k	k	1

SDB stands for spatial downsampling block, BN stands for batch normalization [25], GAP stands for global average pooling, FC stands for the fully connected layer, k denotes the number of classes systems. In this section, we have analyzed various efficient basic architectures.

Iandola et al. [11] proposed SqueezeNet, which introduced three strategies to design a compact model, i.e., replacing 3×3 filters with 1×1 filters, reducing the number of input channels for 3×3 filters, and performing late downsampling in the network to preserve large feature maps. Howard et al. [12] proposed MobileNet, which replaced the majority of 3×3 filters with 1×1 kernels and depth-wise separable convolutions, resulting in a significant reduction in computational cost. Sandler et al. [13] further proposed MobileNetV2. MobileNetV2 introduced residual connections and constructed an inverted residual structure, where the intermediate layer of a block had more channels than its input and output. To maintain representational ability, certain nonlinear functions were removed. Zhou et al. [24] reconsidered the necessity of inverted bottlenecks and suggested that the classic bottleneck structure could also achieve high performance. Based on this, they presented MobileNeXt. Recognizing that 1×1 convolutions contributed significantly to computational cost, ShuffleNet [14] replaced them with group convolutions, employing the channel shuffle operation to facilitate information flow across different groups. By investigating factors that impact practical runtime speed, ShuffleNetV2 [15] introduced a hardware-friendly new block. Han et al. [16] presented GhostNet, which leveraged feature redundancy and replaced half of the channels in 1×1 convolutions with inexpensive operations. Tang et al. [17] presented GhostNetV2, which was characterized by a hardware-friendly attention mechanism called dubbed DFC attention. Its core contribution lay in its ability to capture long-range spatial information while maintaining the implementation efficiency of lightweight CNNs.

In addition to manual design, there has been a range of methods that are aimed at searching for lightweight architectures. E.g., Wu et al. [18] proposed Facebook-Berkeley-Net, which employed a hardware-aware searching strategy that enabled the direct identification of a favorable trade-off between accuracy and speed on specific hardware. Mobile neural architecture search net (MnasNet) [19] and MobileNetV3 [20], based on the inverted residual bottleneck, explored architecture parameters such as model width, model depth, convolutional filter size, and more. While NAS methods have achieved high performance, their success relied on well-designed search spaces and architectural blocks. Combining automatic searching with manual design can lead to the discovery of improved architectures.

Nevertheless, in order to meet the demand for fast inference, it is not feasible to design lightweight CNNs with excessive depth. Most existing CNNs are limited in their ability to capture long-range spatial dependencies due to the inherent limitation of convolutional operations, which only capture local information within a window region [11–20]. Consequently, further performance improvement is impeded. While the combination of self-attention operations and convolutional operations enables the capture of global spatial dependencies, it introduces a computational complexity of $\mathcal{O}(CH^2W^2)$. Compared to most of current lightweight CNNs, our ShuffleNeMt not only maintains computational efficiency, but also captures long-range spatial dependencies. As a result, it exhibits superior performance.

3 Proposed ShuffleNeMt

This section first analyzes the overall framework of ShuffleNeMt. Then, it dissects the downsampling block, ShuffleNeMt block, and local residual block individually. Finally, it introduces the nonlinearity in ShuffleNeMt.

3.1 Overall frameworks

The overall framework of ShuffleNeMt is defined in Table 1. ShuffleNeMt is categorized into four models: ShuffleNeMt-×0.5, ShuffleNeMt-×1.0, ShuffleNeMt-×1.5 and ShuffleNeMt-×2.0. These models are ranked from low to high based on the required resources. Based on the design pattern of ResNet, a spatial downsampling block (e.g., Stacked Spatial Downsampling Block or Stepwise Spatial Downsampling Block) is followed by several skip connection blocks (e.g., Local Residual Block, Local ShuffleNeMt Block, or Non-local ShuffleNeMt Block). The benefits of this design can be summarized as follows. (1) Feature reuse and gradient propagation. Skip connections allow the network to reuse features from shallower layers in deeper layers. This helps the network retain and utilize useful information learned in shallower layers while learning new features. Additionally, skip connections facilitate more efficient backpropagation of gradients, mitigating the vanishing gradient problem and enabling more effective network training. (2) Reducing information loss. During spatial downsampling, some information may be lost due to the reduction in feature resolution. Skip connections can directly transmit high-resolution features to deeper layers of the network, compensating for this information loss to some extent. (3) Improving model performance. Practical experience shows that networks with skip connections often improve model performance without increasing model complexity.

Stacked Spatial Downsampling Block and Stepwise Spatial Downsampling Block are both responsible for downsampling to reduce the spatial resolution of the input and increase the receptive field. Local Residual Block and Local ShuffleNeMt Block capture local positional dependencies through convolutional operators. The Non-local ShuffleNeMt Block captures global positional dependencies through visual self-attention and is used only once in the middle

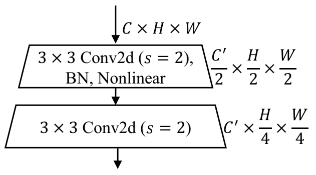

$C \times H \times W$
3×3 Conv2d ($s = 2$), BN, Nonlinear
$\frac{C'}{2} \times \frac{H}{2} \times \frac{W}{2}$
3×3 Conv2d ($s = 2$)
$C' \times \frac{H}{4} \times \frac{W}{4}$

Fig. 2 Stacked spatial downsampling block. s denotes stride, DWConv2d stands for depthwise Conv2d [26]

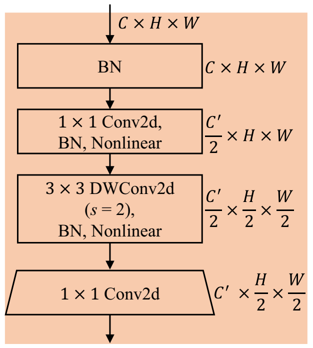

$C \times H \times W$
BN
$C \times H \times W$
1×1 Conv2d, BN, Nonlinear
$\frac{C'}{2} \times H \times W$
3×3 DWConv2d ($s = 2$), BN, Nonlinear
$\frac{C'}{2} \times \frac{H}{2} \times \frac{W}{2}$
1×1 Conv2d
$C' \times \frac{H}{2} \times \frac{W}{2}$

Fig. 3 Stepwise spatial downsampling block. s denotes stride, DWConv2d stands for depthwise Conv2d [26]

of the network. Using this approach, ShuffleNeMt not only maintains its lightweight characteristics but also possesses the ability to capture global positional dependencies.

3.2 Spatial downsampling block

In CNNs, spatial downsampling can reduce the spatial resolution of the input to increase the receptive field. In ShuffleNeMt, there are two types of spatial downsampling blocks: Stacked Spatial Downsampling Block (as shown in Fig. 2) and Stepwise Spatial Downsampling Block (as shown in Fig. 3). The Stacked Spatial Downsampling Block is placed at the beginning of ShuffleNeMt, helping the network quickly reduce the spatial resolution of the input to increase the receptive field. The Stepwise Spatial Downsampling Block is used to gradually reduce the spatial resolution.

3.3 ShuffleNeMt block

Inspired by Full Pre-Activation Units [27], ShuffleNet-V2 block [15], and Vision Transformer [28], we propose a novel skip connection block called ShuffleNeMt Block, as shown in Fig. 4. In Fig. 4, Pre-Norm can be Layer Normalization (LN) [29], BN [25] and etc.. $f_t(.)$ can be a local positional dependency capture unit or a non-local positional dependency capture unit. The scaling path ϵ denotes multiplying $f_t(\text{Pre- Norm}(\boldsymbol{X}_{t_2}))$ by ϵ to scale the parameter gradients, addressing the gradient explosion problem. This design is inspired by DeepNet [30].

Skip connected path scaling The value of ϵ in the current ShuffleNeMt block is set to be $\frac{1}{\sqrt{L}}$, L denotes the depth along the backpropagation direction. E.g., in every ShuffleNeMt block of stage 10, it is set to be 1, $\frac{1}{\sqrt{2}}$, $\frac{1}{\sqrt{3}}$ along the backpropagation direction, respectively. Similarly, in every ShuffleNeMt block of stage 2, it is set to be $\frac{1}{\sqrt{13}}$, $\frac{1}{\sqrt{14}}$, $\frac{1}{\sqrt{15}}$ along the backpropagation direction, respectively.

Local ShuffleNeMt block When $f_t(.)$ is set as the local positional dependency capture unit in Fig. 4, this type of ShuffleNeMt block is named Local ShuffleNeMt Block. Additionally, the Pre-Norm layer is set to BN$(., \boldsymbol{\gamma}, \boldsymbol{\beta})$ by default. The local positional dependency capture unit can be described using Eq. (1),

$$\begin{aligned}
\boldsymbol{Z}_0 &= \text{Pre- Norm}(\boldsymbol{X}_{t_2}, \boldsymbol{\gamma}, \boldsymbol{\beta}),\\
\boldsymbol{Z}_1 &= \delta(\text{BN}(\text{Conv}_{1\times1}(\boldsymbol{Z}_0, \boldsymbol{\Theta}_1), \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1)),\\
\boldsymbol{Z}_2 &= \delta(\text{BN}(\text{DW}_{3\times3}(\boldsymbol{Z}_1, \boldsymbol{\Theta}_2), \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2)),\\
\boldsymbol{Y}_{t_2} &= \text{Conv}_{1\times1}(\boldsymbol{Z}_2, \boldsymbol{\Theta}_3).
\end{aligned} \tag{1}$$

Here $\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2 \in \mathbb{R}^C$, $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2 \in \mathbb{R}^{C\times C\times1\times1}$, $\boldsymbol{\Theta}_3 \in \mathbb{R}^{C\times1\times3\times3}$. $\text{DW}_{3\times3}(.)$ denotes a 3×3 depthwise separable convolution [12], and $\delta(.)$ denotes a nonlinear activation function.

Non-local ShuffleNeMt block When $f_t(.)$ is set as the non-local positional dependency capture unit in Fig. 4, this type of ShuffleNeMt block is named Non-local ShuffleNeMt Block. Additionally, the Pre-Norm layer is set to LN$(., \boldsymbol{\gamma}, \boldsymbol{\beta})$ by default. The core operation of the non-local positional dependency capture unit can be described using Eq. (2),

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{C}}\right)\boldsymbol{V}. \tag{2}$$

Here $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{HW\times C}$. Since CNNs are inherently capable of capturing positional information, the introduction of a visual self-attention unit into CNNs does not require the use of positional encoding [31].

3.4 Local residual block

Due to the output channel number of 24 from the stacked downsampling block, if Local ShuffleNext Blocks are used afterwards, the input channels to $f_t(.)$ would be reduced to only 12. This may lead to insufficient extraction of shallow-level features by the model, thereby compromising performance. Therefore, two local residual blocks are used after the stacked downsampling block, as shown in Fig. 5. The local residual block can be described in detail using Eq. (3),

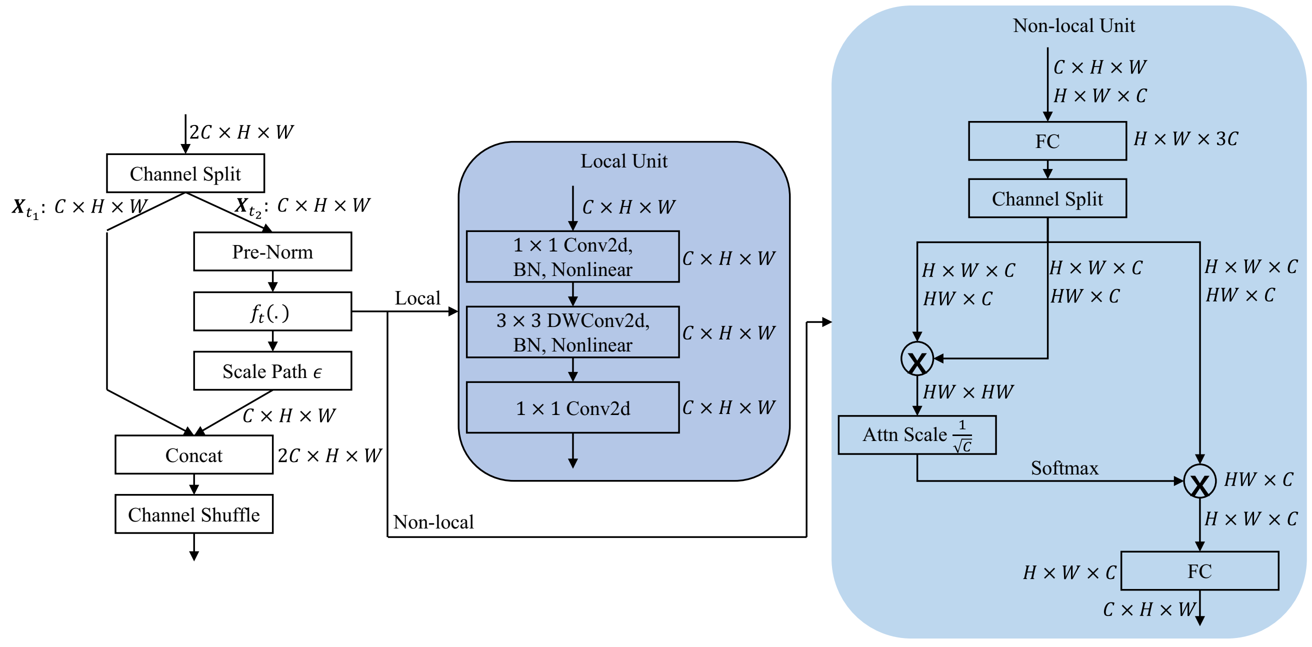


Fig. 4 ShuffleNeMt block

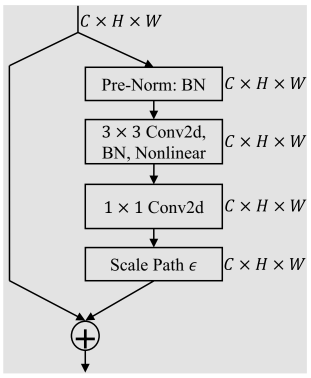


Fig. 5 Local residual block

$$\begin{aligned} \boldsymbol{Z}_1 &= \text{Pre- Norm}(\boldsymbol{X}, \boldsymbol{\gamma}, \boldsymbol{\beta}), \\ \boldsymbol{Z}_2 &= \delta(\text{BN}(\text{Conv}_{3\times3}(\boldsymbol{Z}_1, \boldsymbol{\Theta}_1), \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1)), \\ \boldsymbol{Z}_3 &= \text{Conv}_{1\times1}(\boldsymbol{Z}_2, \boldsymbol{\Theta}_2), \\ \boldsymbol{Y} &= \boldsymbol{X} + \epsilon \boldsymbol{Z}_3. \end{aligned} \tag{3}$$

Here $\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1 \in \mathbb{R}^C$, $\boldsymbol{\Theta}_1 \in \mathbb{R}^{C\times C\times 3\times 3}$, $\boldsymbol{\Theta}_2 \in \mathbb{R}^{C\times C\times 1\times 1}$.

3.5 Nonlinearity

Recently, a series of nonmonotonic activation functions [32–37] have been proposed, demonstrating stable performance that surpasses the Rectified Linear Unit (ReLU) [38]. The reasons for this can be summarized as follows.

1 These nonmonotonic activation functions approximate the identity function $y_i = x_i$ in the positive direction of the x-axis, which helps preserve the original input information. Meanwhile, the derivative in the positive direction of the x-axis is approximately 1, facilitating the stable propagation of gradients for positive inputs during the backpropagation process and preventing issues such as gradient vanishing or gradient explosion caused by variations in the derivative of the activation function.
2 These nonmonotonic activation functions maintain smoothness in the negative direction of the x-axis, contributing to a smoother landscape of the overall loss function. Furthermore, they tend to zero rather than consistently being equal to zero in the negative direction of the x-axis, which helps alleviate the problem of zero gradients for negative inputs and maintains sparsity.

In our ShuffleNeMt, nonlinearity is set to be the Power Function Linear Unit (PFLU) [36], as defined in Eq. (4),

$$\text{PFLU}(x_i) = \frac{x_i}{2}\left(1 + \frac{x_i}{\sqrt{1 + x_i^2}}\right). \tag{4}$$

Figure 6 plots comparison between ReLU and PFLU.

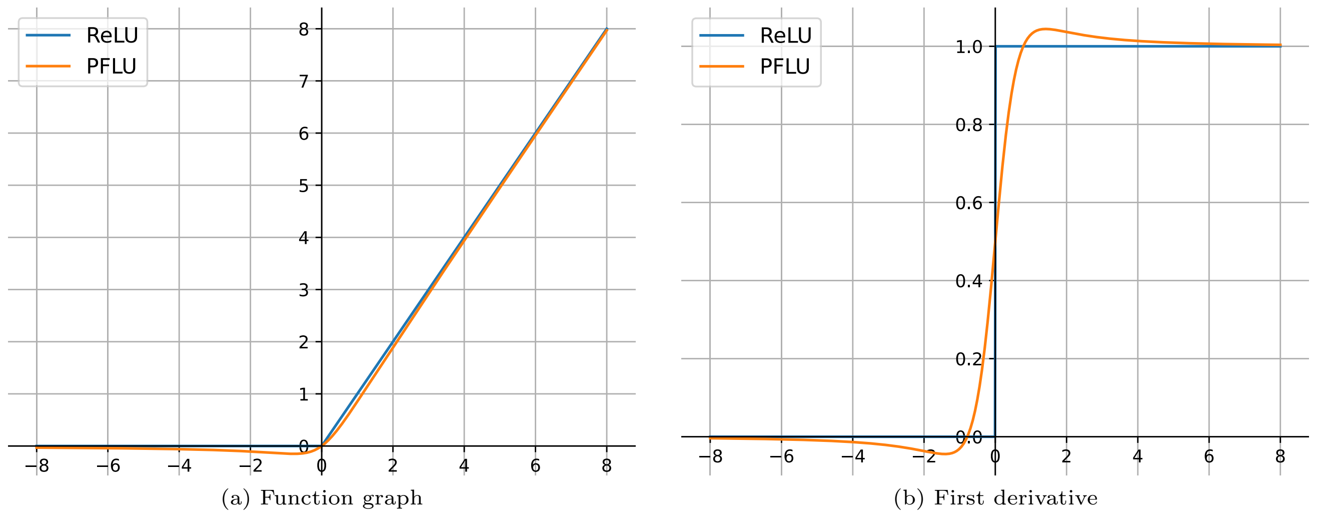


Fig. 6 ReLU vs. PFLU

4 Experiments

In this section, to evaluate the performance of our proposed ShuffleNeMt model, we selected three different visual benchmark tasks: image classification, object detection and semantic segmentation. We conducted experiments using the CIFAR-100 [21], PASCAL VOC [22] and CityScapes [23] datasets, respectively. Simultaneously, we compared the performance of ShuffleNeMt with four popular lightweight CNNs as baseline models, including ShuffleNetV2 [15], MobileNetV3-Large [20], MobileNeXt [24] and GhostNetV2 [17]. Through detailed experimental evaluations, we compared the performance of each model across different tasks.

4.1 Implementation details

To begin with, we conducted evaluation and comparison of various lightweight CNN architectures on the CIFAR-100 dataset [21] (all lightweight CNN architectures only retained the last three spatial downsamplings). Parameters of networks were optimized using the Adam [40] optimizer with a mini-batch size of 128. All models were trained from scratch for 80 epochs. The initial learning rate was set to $2e-3$. If the step value was less than 390, the learning rate scaling factor linearly increased from 0 to 1. For step values between 390 and 21450, the learning rate scaling factor linearly decayed from 1 to 0.01. If the step value was greater than or equal to 21450, the learning rate scaling factor remained constant at 0.01.

Next, we evaluated and compared various lightweight CNN architectures on the PASCAL VOC dataset [22]. The network was trained from scratch for 80 epochs using the Adam optimizer with a mini-batch size of 128. The initial learning rate was set to $4e-3$. If the step value was less than 133, the learning rate scaling factor linearly increased from 0 to 1. For step values between 133 and 7315, the learning rate scaling factor linearly decayed from 1 to 0.01. If the step value was greater than or equal to 7315, the learning rate scaling factor remained constant at 0.01.

Furthermore, we evaluated and compared various lightweight CNN architectures on the CityScapes dataset [23]. The network was trained from scratch for 80 epochs using the Adam optimizer with a mini-batch size of 64. The initial learning rate was set to $2e-3$. If the step value was less than 46, the learning rate scaling factor linearly increased from 0 to 1. For step values between 46 and 2530, the learning rate scaling factor linearly decayed from 1 to 0.01. If the step value was greater than or equal to 2530, the learning rate scaling factor remained constant at 0.01.

In each convolution layer and fully connected layer, the weights were initialized using Xavier uniform distribution [41, 42], and if biases were used, they were initialized with zeros. In each BN layer [25] and LN layer [29], if the scaling parameter (γ) was used, it was initialized with ones, and if the shift parameter (β) was used, it was initialized with zeros.

All performance benchmark tasks were conducted on a single computer with the following specifications: an Intel(R) Core(TM) i7-11700K CPU, Nvidia RTX A5000 GPU, 32GB RAM, Python version 3.11.7, PyTorch [43] version 2.2.1, Cudatoolkit version 12.1 and Cudnn version 8.9.6. All time benchmark tasks were conducted on another single computer with the following specifications: an Intel(R) Xeon(R) E-2136 CPU, Nvidia Quadro RTX 4000 GPU, 16GB RAM, Python version 3.12.2, PyTorch version 2.2.2, Cudatoolkit version 12.1 and Cudnn version 8.9.6.

Table 2 Comparison results between different lightweight CNN models on CIFAR-100 [21]. The **bold** denotes the best result

Model	Param. (M) ↓	CPU (FPS) ↑	GPU (FPS) ↑	Top-1 err. (%) ↓	Top-5 err. (%) ↓
ShuffleNetV2-×0.5 [15]	**0.444**	**124.62**	104.90	43.68	17.44
MobileNetV3-Large-×0.35 [20]	0.593	102.41	**108.94**	41.25	15.66
MobileNeXt-×0.2 [24]	0.568	72.97	87.46	45.00	17.74
GhostNetV2-×0.2 [17]	0.556	49.89	45.59	47.12	19.01
ShuffleNeMt-×0.5 (ours)	0.453	109.15	98.17	**39.98**	**14.84**
ShuffleNetV2-×1.0 [15]	**1.356**	**113.62**	104.05	38.89	14.61
MobileNetV3-Large-×0.55 [20]	1.393	96.47	**108.72**	37.33	13.94
MobileNeXt-×0.7 [24]	1.459	71.64	85.59	37.38	13.35
GhostNetV2-×0.5 [17]	1.705	41.13	45.32	41.05	15.58
ShuffleNeMt-×1.0 (ours)	1.363	93.92	96.42	**35.10**	**11.97**
ShuffleNetV2-×1.5 [15]	**2.581**	**96.91**	105.51	36.90	13.44
MobileNetV3-Large-×0.8 [20]	2.809	85.11	**109.96**	36.86	14.42
MobileNeXt-×1.1 [24]	2.903	62.05	86.74	35.12	12.02
GhostNetV2-×0.7 [17]	2.826	41.53	45.36	38.56	14.14
ShuffleNeMt-×1.5 (ours)	2.588	73.58	97.63	**34.05**	**11.02**

4.2 Image classification

In order to assess and compare the performance of different lightweight CNN model, we conducted experiments on the CIFAR-100 dataset [21]. This dataset comprises colored natural images with dimensions of 32×32 pixels and encompasses 100 categories, each containing 600 images. During the training phase on CIFAR-100, we applied common data augmentation methods including random color space transformations and random horizontal flipping. Our evaluation metrics consisted of the top-1 error rate and top-5 error rate. The results were listed in Table 2.

Optimal performance As shown in Table 2, ShuffleNeMt achieved the lowest top-1 error rate and top-5 error rate under the same order of magnitude parameter complexity.

Computational complexity According to the results in Table 2, ShuffleNetV2 achieved the fastest inference speed on the CPU, MobileNetV3-Large achieved the second fastest inference speed on the CPU, and ShuffleNeMt achieved the third fastest inference speed on the CPU, which was only slightly lower than MobileNetV3-Large, all under the same order of magnitude parameter complexity. Under the same conditions, MobileNetV3-Large achieved the fastest inference speed on the GPU, ShuffleNetV2 achieved the second fastest inference speed on the GPU, and ShuffleNeMt achieved the third fastest inference speed on the GPU, which was only slightly lower than ShuffleNetV2.

4.3 Object detection

We proceeded with experiments on the PASCAL VOC 2007/2012 dataset [22]. This dataset consists of 20 classes. The PASCAL VOC 2012trainval dataset contains 11530 images with 27450 ROI annotated objects. We trained all object detectors using the VOC 2012trainval dataset and evaluated the results on the VOC 2007test dataset for comparison. During the training process on PASCAL VOC, we utilized standard data augmentation techniques, including random color space transformations and random horizontal flipping. The input image resolution for the detectors was set to 320×320 pixels. We employed mAP@50 and mAP@75 as evaluation metrics [44]. The corresponding results were listed in Table 3.

Optimal performance As shown in Table 3, under the same order of magnitude parameter complexity, ShuffleNeMt achieved the highest mAP@50 of 38.36% and the highest mAP@75 of 12.10%. In addition, the highest mAP@50 achieved by ShuffleNeMt, which is 38.36%, was 2.4% higher than the second highest mAP@50 of 35.96%.

Computational complexity Based on the results listed in Table 3, it was evident that ShuffleNetV2 exhibited the quickest inference speed on the CPU, followed by MobileNetV3-Large in second place, and ShuffleNeMt in third, with its speed marginally slower than MobileNetV3-Large. These rankings were consistent across models with similar parameter complexity. In a parallel comparison on the GPU, MobileNetV3-Large demonstrated the fastest inference speed, ShuffleNetV2 came in second, and ShuffleNeMt placed third, closely trailing ShuffleNetV2.

4.4 Semantic segmentation

We further presented additional findings on the well-known CityScapes [23]. This dataset encompasses a range of urban street scenes from over 50 distinct cities captured at different times of the year, and includes ground truth data for

Table 3 Comparison results between different lightweight CNN models on PASCAL VOC 2007test

Model	Param. (M) ↓	CPU (FPS) ↑	GPU (FPS) ↑	mAP@50 (%) ↑	mAP@75 (%) ↑
ShuffleNetV2-×1.0 [15] + YOLOV7 loss [4]	**1.330**	**44.89**	103.06	33.86	9.92
MobileNetV3-Large-×0.55 [20] + YOLOV7 loss [4]	1.375	43.91	**109.60**	35.96	11.26
MobileNeXt-×0.7 [24] + YOLOV7 loss [4]	1.427	30.88	86.02	34.30	11.19
GhostNetV2-×0.5 [17] + YOLOV7 loss [4]	1.673	25.77	45.36	35.67	11.03
ShuffleNeMt-×1.0 (ours) + YOLOV7 loss [4]	1.337	40.96	98.17	**38.36**	**12.10**

"* + YOLOV7 loss" indicates that the original backbone network in YOLOV7 is replaced with a lightweight CNN model *, while changing the multi-scale output to only keeping the output of the last scale, and following the original configuration of YOLOV7 for the rest

Table 4 Comparison results between different lightweight CNN models on CityScapes [23]

Model	Param. (M) ↓	CPU (FPS) ↑	GPU (FPS) ↑	mIOU (%) ↑	Pixel acc. (%) ↑
ShuffleNetV2-×1.0 [15] + FCN loss [39]	**1.274**	**23.53**	103.41	36.37	81.93
MobileNetV3-Large-×0.55 [20] + FCN loss [39]	1.336	20.24	**108.48**	38.08	83.10
MobileNeXt-×0.7 [24] + FCN loss [39]	1.357	12.38	85.99	37.22	82.19
GhostNetV2-×0.5 [17] + FCN loss [39]	1.602	15.11	45.09	36.82	82.43
ShuffleNeMt-×1.0 (ours) + FCN loss [39]	1.281	19.18	97.11	**38.42**	**83.67**

"* + FCN loss" indicates that the original backbone network ResNet in fully convolutional network (FCN) is replaced with a lightweight CNN model *, while not using auxiliary loss, and following the original configuration of FCN for the rest

semantic segmentation and instance-level segmentation. The segmentation labels cover more than 30 classes, we trained our segmentation models using only 20 of them. The basic image resolution input to segmentation models was set to be 520×520. Our evaluation metric was the mean Intersection-Over-Union (mIOU) and pixel accuracy, and our results were listed in Table 4.

Optimal performance As shown in Table 4, under the same order of magnitude of parameter complexity, ShuffleNeMt achieved the highest mIOU of 38.42% and the highest pixel accuracy of 83.67%.

Computational complexity A shown in Table 4, it was apparent that ShuffleNetV2 led with the fastest inference speed on the CPU, followed closely by MobileNetV3-Large in second position, and ShuffleNeMt secured third place with a speed marginally slower compared to MobileNetV3-Large. These observations held true across models possessing comparable parameter complexity. Similarly, on the GPU, MobileNetV3-Large demonstrated superior inference speed, ShuffleNetV2 ranked second, and ShuffleNeMt came in third, with its speed slightly lagging behind ShuffleNetV2.

4.5 Ablation study

To eliminate the influence of nonlinearity on model performance, the nonlinearity in the lightweight CNN models compared in this section is set to ReLU, and ablation experiments were conducted on the CIFAR-100 dataset. Table 5 listed the corresponding ablation experimental results. As shown in Table 5, ShuffleNeMt still stably achieved the lowest top-1 and top-5 error rates under the same order of magnitude of parameter complexity and using the same nonlinearity.

5 Conclusion

In this paper, we propose the ShuffleNeMt model, which aims to achieve efficient performance by reducing parameters and computational complexity, making it suitable for resource-constrained or real-time demanding scenarios, e.g., mobile devices and embedded systems. ShuffleNeMt maintains the lightweight characteristics of the model while gaining the ability to capture global positional dependencies by means of integrating single-layer visual self-attention into a lightweight CNN model. Experimental results demonstrate that ShuffleNeMt outperforms the existing state-of-the-art lightweight CNN models in image classification, object detection, and semantic segmentation tasks, given a similar order of magnitude of parameter complexity. Additionally, ShuffleNeMt exhibits a slightly slower inference speed compared to ShuffleNetV2 and MobileNetV3-Large but is faster than MobileNeXt and GhostNetV2, again assuming a similar parameter complexity.

Future research directions include exploring the application of efficient visual self-attention in ShuffleNeMt to further enhance inference speed with high-resolution inputs, and investigating the optimal combination of ReLU and

Table 5 Ablation comparison results of nonlinearity set as ReLU in different lightweight CNN models on the CIFAR-100 dataset

Model	Param. (M) ↓	Top-1 err. (%) ↓	Top-5 err. (%) ↓
ShuffleNetV2-×0.5 [15]	**0.444**	43.68	17.44
MobileNetV3-Large-×0.35 [20]	0.593	42.37	**15.31**
MobileNeXt-×0.2 [24]	0.568	46.50	19.35
GhostNetV2-×0.2 [17]	0.556	47.12	19.01
ShuffleNeMt-×0.5 (ours)	0.453	**41.73**	15.47
ShuffleNetV2-×1.0 [15]	**1.356**	38.89	14.61
MobileNetV3-Large-×0.55 [20]	1.393	39.61	13.80
MobileNeXt-×0.7 [24]	1.459	37.77	13.10
GhostNetV2-×0.5 [17]	1.705	41.05	15.58
ShuffleNeMt-×1.0 (ours)	1.363	**35.80**	**12.20**
ShuffleNetV2-×1.5 [15]	**2.581**	36.90	13.44
MobileNetV3-Large-×0.8 [20]	2.809	35.89	12.30
MobileNeXt-×1.1 [24]	2.903	34.80	11.83
GhostNetV2-×0.7 [17]	2.826	38.56	14.14
ShuffleNeMt-×1.5 (ours)	2.588	**34.44**	**10.80**

non-monotonic activation functions in ShuffleNeMt using NAS.

Acknowledgements This work was partly supported by the National Natural Science Foundation of China (Grant No. 62076117 and 62166026) and Jiangxi Provincial Key Laboratory of Virtual Reality, China (Grant No. 2024SSY03151).

Data availability The datasets during the current study are open source and available in http://www.cs.toronto.edu/~kriz/cifar.html, http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html and https://www.cityscapes-dataset.com repositories, respectively.

Declarations

Conflict of interest The authors declare no conflict of interest.

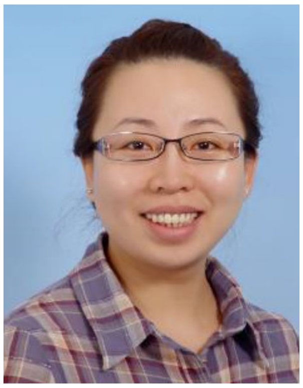
References

1. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp 1097–1105
2. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 770–778
3. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Proceedings of the Advances in Neural Information Processing Systems, p 28
4. Wang CY, Bochkovskiy A, Liao HYM (2023) Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 7464–7475. https://doi.org/10.1109/CVPR52729.2023.00721
5. Karpathy A, Toderici G, Shetty S et al (2014) Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 1725–1732. https://doi.org/10.1109/CVPR.2014.223
6. Szegedy C, Liu W, Jia Y et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 1–9
7. Tan M, Le QV (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: Proceedings of the International Conference on Machine Learning, vol 97. pp 6105–6114
8. Bahdanau D, Cho K, Bengio Y (2014) Neural Machine Translation by Jointly Learning to Align and Translate
9. Gupta A, Anpalagan A, Guan L, Khwaja AS (2021) Deep learning for object detection and scene perception in self-driving cars: survey, challenges, and open issues. Array 10:100057. https://doi.org/10.1016/j.array.2021.100057
10. Liu H, Wu C, Wang H (2023) Real time object detection using lidar and camera fusion for autonomous driving. Sci Rep. https://doi.org/10.1038/s41598-023-35170-z
11. Iandola FN, Moskewicz MW, Ashraf K et al (2016) SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and <1MB Model Size. arXiv:1602.07360
12. Howard AG, Zhu M, Chen B et al (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861
13. Sandler M, Howard A, Zhu M et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 4510–4520
14. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 6848–6856
15. Ma N, Zhang X, Zheng HT, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision. pp 116–131
16. Han K, Wang Y, Tian Q, et al (2020) Ghostnet: More features from cheap operations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 1577–1586. https://doi.org/10.1109/CVPR42600.2020.00165
17. Tang Y, Han K, Guo J et al (2022) Ghostnetv2: Enhance cheap operation with long-range attention. In: Proceedings of the Advances in Neural Information Processing Systems, vol 35. pp 9969–9982

18. Wu B, Dai X, Zhang P et al (2019) Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 10726–10734. https://doi.org/10.1109/CVPR.2019.01099
19. Tan M, Chen B, Pang R et al (2019) Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 2820–2828
20. Howard A, Sandler M, Chu G et al (2019) Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp 1314–1324
21. Krizhevsky A, Hinton GE (2009) Learning Multiple Layers of Features from Tiny Images. https://citeseerx.ist.psu.edu
22. Everingham M, Winn J (2011) The pascal visual object classes challenge 2012 (voc2012) development kit. Pattern Analysis, Statistical Modelling and Computational Learning. Tech Rep 8
23. Cordts M, Omran M, Ramos S et al (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 3213–3223
24. Zhou D, Hou Q, Chen Y et al (2020) Rethinking bottleneck structure for efficient mobile network design. In: Proceedings of the European Conference on Computer Vision. pp 680–697
25. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning, vol 37. pp 448–456
26. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 1251–1258
27. He K, Zhang X, Ren S et al (2016) Identity mappings in deep residual networks. In: Proceedings of the European Conference on Computer Vision. pp 630–645
28. Dosovitskiy A, Beyer L, Kolesnikov A et al (2020) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929
29. Ba LJ, Kiros JR, Hinton GE (2016) Layer Normalization. arXiv:1607.06450
30. Wang H, Ma S, Dong L et al (2022) DeepNet: Scaling Transformers to 1,000 Layers. arXiv:2203.00555
31. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: Proceedings of the Advances in Neural Information Processing Systems, vol 30. pp 5998–6008
32. Hendrycks D, Gimpel K (2016) Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. arXiv:1606.08415
33. Ramachandran P, Zoph B, Le QV (2018) Searching for activation functions. In: Proceedings of the International Conference on Learning Representations
34. Ying Y, Su J, Shan P et al (2019) Rectified exponential units for convolutional neural networks. IEEE Access 7:101633–101640. https://doi.org/10.1109/ACCESS.2019.2928442
35. Zhu H, Zeng H, Liu J, Zhang X (2021) Logish: a new nonlinear nonmonotonic activation function for convolutional neural network. Neurocomputing 458:490–499. https://doi.org/10.1016/j.neucom.2021.06.067
36. Zhu M, Min W, Wang Q et al (2021) Pflu and fpflu: two novel non-monotonic activation functions in convolutional neural networks. Neurocomputing 429:110–117. https://doi.org/10.1016/j.neucom.2020.11.068
37. Wang X, Ren H, Wang A (2022) Smish: a novel activation function for deep learning methods. Electronics 11(4):550. https://doi.org/10.3390/electronics11040540
38. Hahnloser RHR, Sarpeshkar R, Mahowald MA et al (2000) Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature 405(6789):947–951
39. Shelhamer E, Long J, Darrell T (2017) Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell 39(4):640–651. https://doi.org/10.1109/TPAMI.2016.2572683
40. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Machine Learning
41. Barr DR, Sherrill ET (1999) Mean and variance of truncated normal distributions. Am Stat 53(4):357–361. https://doi.org/10.1080/00031305.1999.10474490
42. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, vol 9. pp 249–256
43. Paszke A, Gross S, Massa F et al (2019) Pytorch: An imperative style, high-performance deep learning library. In: Proceedings of the Advances in Neural Information Processing Systems. pp 8026–8037
44. Padilla R, Netto SL, da Silva EAB (2020) A survey on performance metrics for object-detection algorithms. In: Proceedings of the International Conference on Systems, Signals and Image Processing. pp 237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130

Meng Zhu is currently pursuing the Ph.D. degree at Nanchang University, China. He obtained the B.E. and M.E. degrees in computer science and technology from Nanchang University, China in 2021 and 2018, respectively. His current research interests are basic architectures of artificial intelligence, multimodal image fusion, security of large models, and security of agents. He is a Reviewer of *EAAI*.

Weidong Min received the B.E., M.E. and Ph.D. degrees in computer application from Tsinghua University, China in 1989, 1991 and 1995, respectively. He is currently a full Professor of School of Mathematics and Computer Science, Nanchang University, China, the Dean of Institute of Metaverse, Nanchang University, China, the Director of Jiangxi Provincial Key Laboratory of Virtual Reality, Jiangxi, China, and an Executive Director of China Society of Image and Graphics. His current research interests include image and video processing, virtual reality, metaverse, artificial intelligence and big data.



Qing Han received the B.E. and M.E. degrees in computer application from Tianjin Polytechnic University, China in 1997 and 2006, respectively. She is currently an Associate Professor of School of Mathematics and Computer Science, Nanchang University, China. Her current research interests include image and video processing, network management, virtual reality and metaverse.

Guowei Zhan received the M.E. degree in computer science and technology from Nanchang University, China in 2023. Her current research interests include computer vision, and vehicle re-identification.

Qiyan Fu received the M.E. degree in Electronic and Communication Engineering from the Nanchang University, China in 2017. She received the Ph.D. degree from the Nanchang University, China in 2024. Her current research interests focus on artificial intelligence, and computer vision.

Jiahao Li received the B.S. degree in information science and technology from Shandong University of Technology, China in 2020. He received the M.E. degree in computer science and technology from Nanchang University, China in 2023. He is currently pursuing the Ph.D. degree at Nanchang University, China. His current research interests include computer vision, digital twins, digital human and virtual reality.