Received 3 August 2024, accepted 16 August 2024, date of publication 21 August 2024, date of current version 2 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3447018

APPLIED RESEARCH

VarKFaceNet: An Efficient Variable Depthwise Convolution Kernels Neural Network for Lightweight Face Recognition

QINGHUA MA, (Graduate Student Member, IEEE), PENG ZHANG, AND MIN CUI
School of Instrument and Electronics, North University of China, Taiyuan 030051, China

Corresponding author: Peng Zhang (zhangpeng6@nuc.edu.cn)

This work was supported in part by the National Defence Fund under Grant 2023-JJ-0353, and in part by the National Natural Foundation of China under Grant 62373247.

ABSTRACT We revisit the design of convolutional kernels in lightweight convolutional neural networks, and inspired by the recent advances in RepLKNet, we design a Variable Kernel Convolutional Network module VarKNet, which solves the problem of the imbalance between depthwise convolution and pointwise convolution in the case of depthwise separable convolution when the network width is large, and enriches the model's receptive field. The VarKNet module adopts a multi-branch structure during training and is re-parameterized and fused into a single-path structure during inference to maintain the strong expressive ability of the model and improve the inference speed. In order to further enhance the information exchange between channels, VarKNet adds channel shuffling in the fused branches. Built on VarKNet, we designed a large-scale face recognition network VarKFaceNet. VarKFaceNet achieved A great achievement of 99.5% accuracy on the LFW dataset with 0.7M parameters and 0.24 GFLOPS. At the same time, the measured speed on the NVIDIA Jetson Nano platform is 159 times, 4.2 times, and 2.4 times that of ResNet-50, EfficientNet, and MobileFaceNet, respectively. VarKFaceNet excels in balancing speed and accuracy and is quite suitable for embedded devices with limited resources.

INDEX TERMS Face recognition, local features, multi-scale, lightweight network.

I. INTRODUCTION

The methodology of deep learning representation is to use simple modules to realize highly complex function representations [1], and Convolutional Neural Networks (CNNs) are one of the most effective methods for deep learning representation. CNNs have achieved significant success in downstream tasks such as face recognition. However, implementing large-scale face recognition CNN models for rapid inference on resource-constrained embedded devices remains a substantial challenge.

Numerous studies have focused on building efficient lightweight convolutional neural networks, such as low-rank matrix factorization [2], which operates on a fully connected layer to reduce the memory requirements and complexity of the model. HashNet [3] introduces a method to effectively quantize network weights. Before the training process begins, the network weights are hashed into distinct groups, with the weights within each group being shared. This approach means that only the shared weights and hash indexes need to be stored, resulting in substantial memory savings. Additionally, weight pruning [4] is employed to retain only the essential connections, thereby reducing the memory footprint and computational requirements of the neural network by an order of magnitude, all while preserving its accuracy. Knowledge Distillation [5] achieves model compression by training shallow models guided by pre-trained Teacher Models and using the output probabilities of the pre-trained models as soft labels for the new shallow models. The aforementioned works have significantly advanced the application of deep learning on mobile and embedded devices. However, their outstanding performance still heavily relies on the inherent capabilities of the neural network models themselves.

In addition, some researchers have focused on building more efficient models through the development of more effective CNN modules, yielding significant results.

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

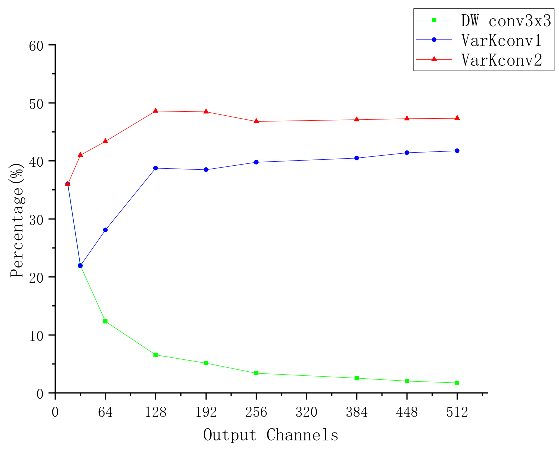

FIGURE 1. Comparison of the proportional computational load of depthwise convolution for DW Conv3 × 3, VarKNet1, and VarKNet2 across various channel widths. The x-axis represents the number of output channels, while the y-axis shows the percentage of total computational load contributed by depthwise convolution.

SqueezeNet [7] uses Conv1 × 1 to reduce the dimensionality of the feature map, thereby reducing the number of weighting parameters. MobileNet [8] relies on depthwise convolutional operations and inverse residual structures to construct efficient modules with competitive performance. ShuffleNet [9] uses channel shuffling operations, and Vision Transformer (ViT) [10] applies the Transformer to images for the first time by segmenting the image and then feeding it into the Transformer network after position and content encoding, achieving great success in visual tasks such as image categorization and representation learning. RepLKNet [11] achieved comparable or better performance than Swin [13] with a 31 × 1 kernel convolutional network and structural re-parameterization [14], while having a faster inference speed. The above work achieved excellent performance in both image classification and recognition, but the optimization problem on embedded systems still exists with embedded hardware and corresponding compilers [15]. VarGNet [15] proposes a variable group convolution, which mitigates the imbalance of computational intensity within a block to some extent. However, this work fails to point out that network width is the most critical factor influencing the amount of depthwise separable convolutional computation, focusing on pointwise convolution, as well as the fact that multibranching also has a significant impact on latency. Inspired by RepLKNet and VarGNet, we revisit the strategy of using large kernel convolution in lightweight networks and cleverly set up the variable kernel convolution module VarKNet according to the number of channels to address the computational imbalance and inefficient feature extraction caused by small kernel depthwise separable convolution when the network width is large.

As shown in Figure 1, when the number of output channels is 384, depthwise convolution accounts for only 2.3% of the computation, while pointwise convolution consumes 97.7%. This results in a significant imbalance of computation within a block, leading to inefficiency in feature extraction. This problem is further exacerbated when neural network inference is performed on embedded devices.

To address this issue, we propose an efficient module called VarKNet, which increases the convolution kernel size based on network width, effectively mitigating the imbalance. As shown in Figure 1, we employed two different strategies to configure VarKNet, both of which successfully alleviate the computational imbalance between depthwise and pointwise convolutions. To further enhance performance, VarKNet incorporates shortcut connections. Additionally, we implemented structural re-parameterization techniques and channel mixing operations.

To validate the performance of VarKNet, we built a lightweight face recognition network named VarKFaceNet. VarKFaceNet was evaluated on several datasets, including LFW, VGG2, and CPLFW. Using the LFW dataset as an example, VarKFaceNet achieved an accuracy of 99.5%, sacrificing only 0.1% accuracy compared to MobileNetV2 [16], while achieving a 4-fold reduction in parameters and a 6-fold reduction in FLOPs. This performance is particularly impressive considering the use of large kernel convolutions, such as DW Conv13 × 13.

Our contribution is as follows:

1) A comprehensive analysis of the causes of intra-block computational imbalance was conducted, leading to the development of a novel lightweight modular variable kernel convolution, VarKNet. VarKNet not only addresses the contradiction of intra-block computational imbalance in embedded devices but also possesses powerful feature extraction capability, representing a novel design paradigm for lightweight convolutional neural networks (CNNs).

2) A dedicated lightweight face recognition network, VarKFaceNet, was designed based on VarKNet. VarKFaceNet achieves a balance between speed and accuracy, as evidenced by its excellent test results on LFW, VGG, and other datasets.

3) Structural re-parameterization techniques were employed to utilize multiple branches during training to extract features with different receptive fields. During inference, these branches are converted into a single-path structure to accelerate inference. Additionally, channel mixing operations were introduced to enhance inter-channel communication in depthwise separable convolutions.

II. RELATED WORK

This section provides a concise overview of select key lightweight convolutional neural network designs and related work in the field of lightweight face recognition-specific networks.

A. LIGHTWEIGHT FACE RECOGNITION NETWORKS

Previously, approaches to lightweight models have focused on the design of simpler network structures, the reduction of the number of layers or channels, or the compression of parameters. However, these methods typically lead to a compromise in model performance. MobileNet represents a novel approach to convolutional neural networks, proposing the replacement of conventional convolution with depthwise separable convolution. This approach offers the possibility of significantly reducing the amount of computation while maintaining model performance. Furthermore, MobileNetV2 incorporates an inverted residual structure and a linear bottleneck layer, which enhances the model's structural fairness and efficiency. MobileNetV3 [17] employs complementary search techniques to combine these modules into a more efficient model. ShuffleNet employs group convolution and channel shuffling operations, while ShuffleNetV2 [18] achieves efficient information exchange and feature learning through appropriate channel rearrangement and depthwise separable convolutions. GhostNet v2 [19], based on GhostNet, incorporates decoupled fully connected attention to enhance its representation capability, thereby capturing long-range dependencies between different spatial pixels. It is a novel lightweight convolutional neural network module. Partial Convolution (PConv) [20] enhances the efficiency of spatial feature extraction by reducing redundant computations and memory access. Specifically, it performs convolution only on a subset of channels, while the remaining channels are left unprocessed and concatenated with the convolution results to form the output, rather than being added. DenseNet [12] employs a densely connected approach to reuse feature information, thereby achieving enhanced performance with reduced computational effort. RepVGG [14] incorporates the residual connection structure of ResNet, drawing upon the principles of simple convolutional stacking and channel attention from VGGNet. It employs a multi-branch network for training and a VGG-like vertical network for inference, thereby achieving an optimal balance between speed and accuracy.

B. LIGHTWEIGHT SPECIALIZED NETWORK FOR FACE RECOGNITION

Despite the excellent results achieved by lightweight models, the design of dedicated face recognition convolutional neural networks for resource-constrained embedded devices remains a problem. MobileFaceNet [21] and ShuffleFaceNet [22] are built on the MobileNetV2 and ShuffleNetV2 designs, respectively, and have achieved significant accuracy. MobileFaceNet introduces a global depthwise convolution layer that replaces the traditional global average pooling. This design reduces the number of parameters and computational cost while maintaining high efficiency, making it suitable for mobile and embedded devices. MobiFace [23] extends MobileNetV2 by incorporating an efficient attention module and feature fusion strategy, which significantly improves the representational capability and performance of the model across different poses and ages, while keeping the computational cost low. These approaches improve the performance of lightweight face recognition networks by optimizing the network architecture and incorporating attention mechanisms. MobileFaceNet focuses on reducing computational resources, while MobiFace improves the robustness and adaptability of the model at low computational cost. Depthwise separable convolution using small kernels has become the standard approach for lightweight CNN design and has yielded satisfactory results. Nevertheless, the optimization problem for embedded systems is still a significant challenge. A novel variable convolutional kernel network module, VarKNet, has been established to address the computational imbalance within a lightweight model block due to the use of depthwise separable convolution in resource-constrained embedded or mobile devices. A further network, VarKFaceNet, was also based on VarKNet, which is a modern lightweight face recognition-specific network.

III. PROPOSED METHOD

In this section, we first introduce our novel, efficient convolutional neural network VarKNet, which has been designed to fulfill the two criteria for variable kernel convolution. Secondly, we present the new, specialized network VarKFaceNet for face recognition, including the network architecture, activation function, and loss function.

A. VarKNet: VARIABLE KERNEL CONVOLUTION MODULE

1) VarKNet: VARIABLE KERNEL CONVOLUTION MODULE

Convolution methods that adhere to traditional conventions necessitate the processing of the feature map in a channel-by-channel fashion. This process is inherently computationally expensive, with a total computational effort that is illustrated in (1). Consequently, a substantial body of research has been conducted with the objective of developing more efficient convolution methods. One such method is Group Convolution [24], which divides the input data into g groups with the number of groups specified by g. Consequently, the number of input channels per convolution kernel is reduced to c_i/g of the number of input channels. Subsequently, the results of each convolutional group are spliced together to form the final output, with the number of output channels of the convolutional kernel becoming c_{i+1}/g. The complete calculation process is illustrated in (2). In comparison to the conventional convolution operation, the computational complexity of the group convolution is reduced by a factor of $1/g$. The concept of Depthwise Separable Convolution was first introduced by the Google team in MobileNet, where a convolution kernel is responsible for only one channel. Subsequently, the width of the network was extended and inter-channel communication was enhanced by Pointwise Convolution. The formulas in equations (3) and (4) represent the calculations for depthwise convolution and pointwise convolution, respectively. The computational cost of depthwise separable convolution is reduced by approximately $1/k^2$ compared to conventional convolution, as demonstrated in (5). This reduction greatly facilitates the deployment of neural networks on embedded devices.

$$\mathrm{MACs}_{\mathrm{standard}} = h_i \times w_i \times c_i \times c_{i+1} \times k \times k \tag{1}$$

$$\mathrm{MACs}_{\mathrm{group}} = \frac{h_i \times w_i \times c_i \times c_{i+1} \times k \times k}{g} \tag{2}$$

$$\mathrm{MACs}_{\mathrm{depth}} = h_i \times w_i \times c_i \times k \times k \tag{3}$$

$$\mathrm{MACs}_{\mathrm{point}} = h_i \times w_i \times c_i \times c_{i+1} \tag{4}$$

$$\mathrm{MACs}_{\mathrm{depthwise}} = \mathrm{MACs}_{\mathrm{depth}} + \mathrm{MACs}_{\mathrm{point}} \tag{5}$$

$$\text{Percentage}_{\text{depth}} = \frac{k^2}{k^2 + c_{i+1}} \times 100 \tag{6}$$

$$k = \{k \,|\, k^2 < c_{\text{out}}, k \text{ is odd}\} \quad (k = 3, 5, 7, \ldots) \tag{7}$$

$$k = \{k \,|\, (k+2)^2 < c_{\text{out}}, k \text{ is odd}\} \quad (k = 3, 5, 7, \ldots) \tag{8}$$

where:

- h_i is the height of the input feature map.
- w_i is the width of the input feature map.
- c_i is the number of input channels.
- c_{i+1} is the number of output channels.
- k is the kernel size.
- g is the number of groups.

According to (6), when the size of the input and output feature maps are kept constant, the difference in the computational volume between pointwise convolution and depthwise convolution is mainly affected by the size of the convolution kernel and the number of output channels. Therefore, we propose a new dynamic variable kernel convolution, VarKConv. The variable kernel convolution is demonstrated in Fig. 2, where we dynamically expand the size of the convolution kernel as the width of the network is increased in order to balance the computational effort between depthwise convolution and pointwise convolution. Depending on the expansion speed of the convolution kernel size, we propose two criteria as shown in (7) and (8). The latter criterion puts a brake on the expansion speed of convolutional kernels and is more suitable for lightweight networks. We evaluate the share of depthwise convolution in the total computation for DW Conv3 × 3, variable kernel convolution guided by Guideline 1, and Guideline 2 when the number of output channels is expanded from 0 to 512. As shown in Fig. 1, the share of depthwise convolution computation for the traditional small kernel DW Conv3 × 3 decreases rapidly when the width of the network is expanded, while the variable kernel convolution guided by Guideline 1 accounts for approximately 40% to 50% of the total computation, and Guideline 2 accounts for approximately 35% to 45%. By increasing the convolution kernel size, we can obtain a larger receptive field, which greatly improves feature extraction. Therefore, it is not necessary to increase the number of channels excessively to achieve good performance, which is one of the reasons why we increase the convolution kernel size without incurring unaffordable computational costs.

2) VarKNet: VARIABLE KERNEL CONVOLUTION MODULE

The VarKNet module is mainly designed for lightweight networks, which are typically multi-branch networks. It includes a main branch consisting of variable kernel convolution, feature extraction branches with different scales, identity mapping, and a hyperparameter that can control the number of layers. As shown in Fig. 3(a), when the kernel size of the main branch convolution is 3, the kernel size of our feature extraction branch convolution is 1 × 1, which is mainly used to pre-extract more local information. As shown in Fig. 3(b), when the kernel size of the main branch

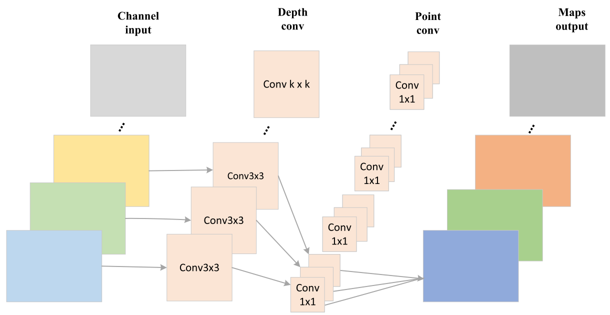

FIGURE 2. Schematic of variable kernel convolution. This diagram illustrates the flow and interactions of different convolution operations in a variable kernel setup. Beginning with channel input, the process involves depthwise convolution, followed by multiple parallel 1 × 1 point convolutions. As the number of channels increases, the sizes of the depthwise convolution kernels are dynamically adjusted according to criteria 1 and criteria 2, culminating in the output of feature maps.

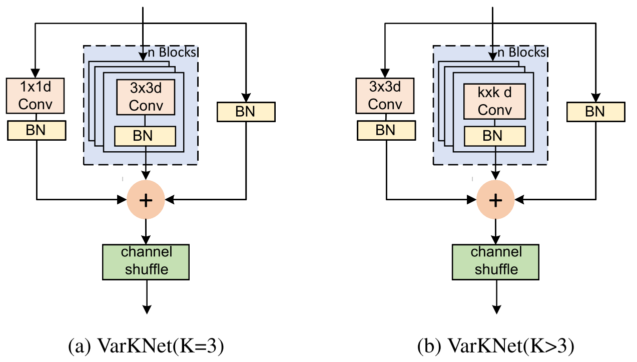

FIGURE 3. Architecture of VarKNet with different kernel sizes. (a) Demonstrates a configuration where the main convolution branch has a kernel size of 3, utilizing 1 × 1 convolutions in the feature extraction branch for local information pre-extraction. (b) Shows a setup where the main convolution branch uses larger than 3 × 3 kernels, complemented by 3 × 3 DW convolutions in the feature extraction branch to enhance the network's receptive field and overall feature extraction capability.

convolution is larger than 3 × 3, our feature extraction branch is set to DW Conv3 × 3. This combined receptive field greatly improves the network's enhancement. The added DW Conv3 × 3 feature extraction branch reintroduces the problem of imbalance between depthwise convolution and pointwise convolution computation. To address this, we use the structural re-parameterization technique, as illustrated in Fig. 4. First, we perform the batch normalization (BN) operation as shown in (9) and (10), and the convoluted BN can be considered as a special convolution. Then, we perform the structural re-parameterization, where the 3 × 3 convolution can be filled into 7 × 7, and the shortcut can first be equated to the special Conv1 × 1 and then expanded into Conv7 × 7. Finally, according to the additivity of convolution, we merge the three branches into a single path structure, achieving a perfect implementation of multi-scale large kernel convolutional neural network fast inference. The biggest difference between DW convolution and ordinary convolution is the inter-channel information communication, so we introduce the channel blending operation proposed in ShuffleNet to further bridge this gap.

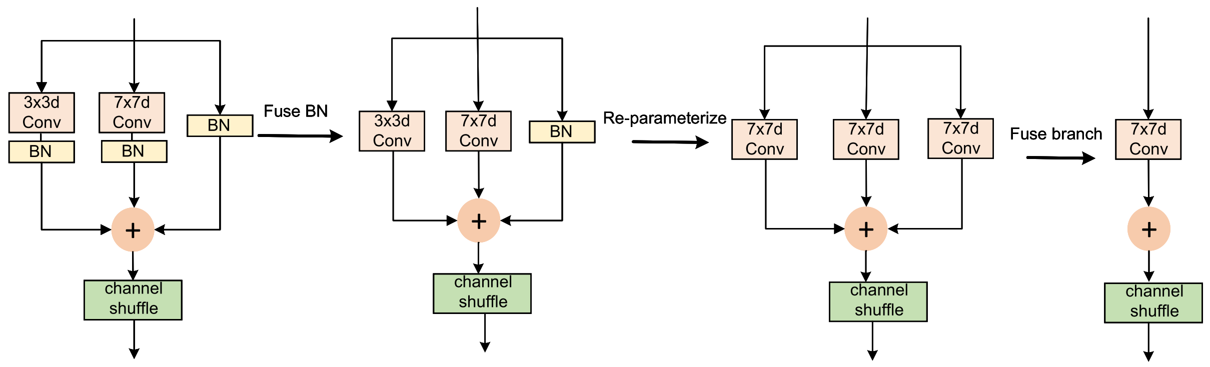

FIGURE 4. Re-Parameterization process of the VarKNet architecture. This figure illustrates the technique of structural re-parameterization used to address the computational imbalance between deep and pointwise convolutions. Initially, BN operations compress batch normalization layers into convolutions, followed by structural re-parameterization where 3 × 3 convolutions are expanded into 7 × 7 formats. Shortcut paths are initially equivalent to special 1 × 1 convolutions and then expand to 7 × 7, enabling integration of three branches into a single streamlined path. This approach facilitates fast inference in a multi-scale large kernel convolutional network.

$$BN(Conv(x)) = x \cdot \frac{W\gamma}{\sigma} + \beta - \frac{\gamma\mu}{\sigma} \tag{9}$$

$$BN(conv(x)) = W_{\text{fused}}(x) + B_{\text{fused}} \tag{10}$$

where:

- x: Input data
- W: Convolution kernel weights
- γ: Trainable scale parameter
- β: Trainable shift parameter
- σ: Standard deviation
- μ: Mean
- W_{fused}: Fused weight after batch normalization and convolution
- B_{fused}: Fused bias after batch normalization and convolution

B. VarKKFaceNet: VARIABLE KERNEL CONVOLUTION MODULE

VarKFaceNet is a lightweight face recognition model based on VarKNet. Its network configuration is detailed in Table 3, and the more intuitive network structure is illustrated in Figure 5. To evaluate the efficacy of the VarKNet module, we constructed the network primarily by stacking VarKFaceNet modules. According to the criteria proposed by ShuffleNetV2, the input and output channels are kept as consistent as possible, altering the number of channels only during downsampling. Furthermore, the size of the convolution kernel in the backbone branch of the VarKNet module is dynamically adjusted in response to changes in the number of channels, as specified by Criterion 1 (7). In the final stage, the network output is embedded into a 128-dimensional face vector through a fully connected layer.

Specifically, the initial layer is a conventional 3 × 3 convolution layer with 16 output channels. The second and third layers are VarKNet (3 × 3) modules, used to extract fine features. The fourth layer is VarKNet (5 × 5) with a stride set for downsampling, increasing the number of channels to 32. The fifth and sixth layers are VarKNet (5 × 5) modules, with the output channels remaining at 32. The seventh to ninth layers follow the same configuration strategy, stacking VarKNet (7 × 7) modules, resulting in 64 output channels. The tenth to twelfth layers are VarKNet (11 × 11) modules with 128 output channels. The thirteenth layer is VarKNet (13 × 13), where the feature map size has been reduced to 7 × 7, with 192 output channels; therefore, the kernel size of the main branch is not expanded further. Finally, a depthwise separable convolution is used to expand the input feature map to 1024 dimensions, followed by a linear Conv1 × 1 operation to produce a 128-dimensional feature vector.

Additionally, to avoid excessive computational demands, the expansion of the convolution kernel can be slowed down. According to Criterion 2(8), the V2 version, as shown in Table 3, can be derived.

TABLE 1. The overall framework of VarKFacenet.

input	operator		n	s	Output
	V1	V2			
112 × 112	Conv 3 × 3	Conv 3 × 3	1	1	16
112 × 112	VarKnet (3x3)	VarKnet (3x3)	2	1	16
112 × 112	VarKnet (5x5)	VarKnet (5x5)	1	2	32
56 × 56	VarKnet (5x5)	VarKnet (3x3)	2	1	32
56 × 56	VarKnet (7x7)	VarKnet (5x5)	1	2	64
28 × 28	VarKnet (7x7)	VarKnet (5x5)	2	1	64
28 × 28	VarKnet (11x11)	VarKnet (9x9)	1	2	128
14 × 14	VarKnet (11x11)	VarKnet (9x9)	2	1	128
14 × 14	VarKnet (13x13)	VarKnet (11x11)	1	2	192
7 × 7	Conv1x1	Conv1x1	-	1	1024
7 × 7	Linear1x1	Linear1x1	-	1	128

C. ACTIVATION AND LOSS FUNCTION

In early neural networks, the sigmoid activation function was widely used as the activation function for each layer, and its expression is determined by (11). When the input data is large or small, the gradient of the sigmoid activation function tends to approach zero. When the gradients used to update the neural network weights become very small,

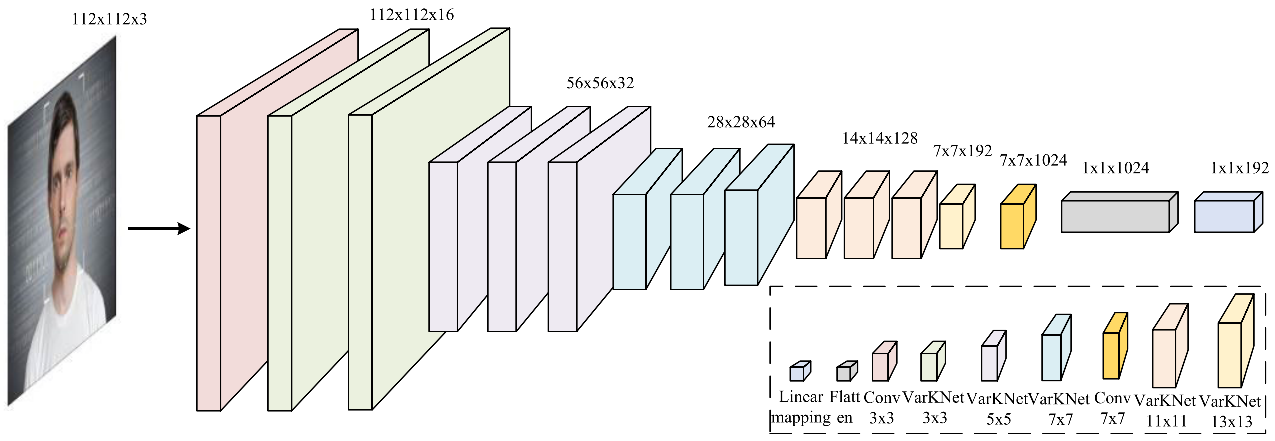

FIGURE 5. The architecture of VarKFaceNet. The model starts with a 3 × 3 convolution layer followed by multiple VarKNet modules with increasing kernel sizes, downsampling at specific layers, and ending with a linear 1 × 1 convolution to produce a 128-dimensional feature vector for face recognition. Different colors of the feature maps represent different operations, indicating the output after each corresponding operation.

the backpropagation algorithm cannot proceed effectively, thereby preventing the network from learning and updating its weights. This phenomenon is known as the gradient vanishing problem. The ReLU function is a take-max function and is the most commonly used activation function, including in many face recognition networks. Its mathematical form is given in (12). Since the ReLU output is zero when the input is negative, which may result in some neurons never being activated, the PReLU can adaptively learn to correct the parameters of the linear units and can improve accuracy with a negligible increase in additional computational cost [25]. Its expression is shown in (13). The face has more symmetric and inverse features, and by allowing negative outputs, the activation function can create larger separation distances in the feature space, thus improving the performance of the classifier. Therefore, we use PReLU as the activation function for VarKFaceNet.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{11}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{12}$$

$$\mathrm{PReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{13}$$

The selection of a loss function is of the utmost importance in the design of lightweight networks. The current classification of loss functions for face authentication can be broadly classified into two categories: classification-based loss functions and metric learning-based loss functions. Classification-based loss functions are sensitive to the issue of mismatch during training and testing, whereas metric learning-based loss functions are prone to the challenge of identifying suitable sampling methods [26]. The most commonly employed cosine Softmax cross-entropy loss function is shown in (14). Since the Softmax function does not specify the intra-class variance, and the face recognition task is not entirely relevant, the ArcFace function reduces the intra-class distance between the sample and the sample center in order to increase the additive angular margin in the loss function, thereby increasing the angular contribution. This makes the optimization process with the angle smaller, thus reducing the intra-class distance, increasing the inter-class spacing, and making it a more suitable function for face recognition tasks. The expression is shown in (15). When the ArcFace loss function is employed directly to train the lightweight convolutional neural network, it is susceptible to two potential issues. Firstly, the loss value may diverge, resulting in suboptimal performance. Secondly, the loss value may fail to converge to a small value, which similarly affects the model's efficacy. Consequently, the model is initially trained with a Softmax loss function, after which the pre-trained Softmax model is trained with ArcFace loss to reduce the intra-class variance and expand the inter-class variance.

$$L_{Softmax} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \tag{14}$$

$$L_{Arc} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos(\theta_j)}} \tag{15}$$

where:

- N is the number of training samples.
- n is the total number of classes.
- x_i is the feature vector of the i-th sample.
- W_{y_i} and W_j are the weight vectors of the correct class and class j, respectively.
- b_{y_i} and b_j are the bias terms of the correct class and class j, respectively.
- θ_{y_i} is the angle between x_i and W_{y_i}.
- θ_j is the angle between x_i and W_j.
- m is the angular margin added to θ_{y_i}.
- s is a scaling factor applied to the cosine value.
- y_i is the ground-truth class of the i-th sample.

IV. EXPERIMENT AND DISCUSSION

This section provides details of the experiments, including the training dataset, test dataset, evaluation dataset, and experimental parameter settings. Additionally, a comparison between VarKFaceNet and other prominent models from recent years is presented. The model performance is then verified on the IJBB and IJBC datasets and compared with benchmarks. Finally, we conducted inference speed tests on the NVIDIA Jetson Nano embedded system.

A. EXPERIMENTAL PARAMETER SETTINGS

1) HARDWARE PLATFORM

The training platform is a Dell high-performance workstation equipped with an NVIDIA GeForce RTX 3090 (24GB) graphics card and an Intel(R) Core(TM) i9-10980XE processor. The specific configurations are shown in Table 2.

TABLE 2. Server hardware configuration.

Server Hardware	Hardware Configuration
CPU	Intel Core i9-10980XE
GPU	NVIDIA GeForce RTX 3090 (24GB)
Memory	256GB
Hard Disk Drive	256GB
Operating System	Windows 10 Pro, version 22H2

2) DATASET SETTINGS

We selected a large dataset, MS1MV3 [27], as the training set, which includes 5.1 million face images and 93,000 labels. The test set consists of five mainstream datasets: LFW [28], CPLFW, CALFW, CFP [29], and VGGFace2-FP [30]. The LFW (Labeled Faces in the Wild) dataset is one of the most widely used benchmarks for face recognition, serving both as a validation and evaluation set. Despite many algorithms achieving high accuracy on this dataset, it remains a crucial benchmark for assessing model performance. The CALFW (Cross-Age Labeled Faces in the Wild) dataset evaluates the robustness of face recognition algorithms to age variations, as it includes cross-age pairs, making it a more challenging benchmark for age differences. Similarly, the CPLFW (Cross-Pose Labeled Faces in the Wild) dataset extends LFW to evaluate performance under varying poses, containing pairs of images with large pose differences, thus testing the algorithm's ability to recognize faces under different orientations. The CFP (Celebrities in Frontal-Profile) dataset assesses the algorithm's robustness to pose variations by containing images of celebrities in frontal and profile views. The VGGFace2-FP dataset is used for frontal-profile face verification, challenging the algorithm's robustness to extreme pose variations. Performance evaluation tests were conducted on the IJBB and IJBC datasets [31], which are widely used benchmarks for large-scale face recognition. The specific configurations are shown in Table 3.

TABLE 3. Dataset information.

Type	Dataset	Label	Images
Training Dataset	MS1M	93k	5.1M
Validation Dataset	LFW	8.6k	3.1M
	CPLFW	5,749	12,174
	CALFW	5,749	11,652
	CFP	500	7,000
	VGGFace2-FP	500	173k
Evaluation Dataset	IJB-B	1,845	76.8k
	IJB-C	3,531	148.8k

3) TRAINING STRATEGY

VarKFaceNet is built on the PyTorch implementation and its output size is a 128-dimensional vector. The batch size is set to 256, the learning rate is determined using the cosine strategy, the warm-up is set to 0.01 for five batches, the initial learning rate is set to 0.1, the momentum is set to 0.9, and the weight decay is set to 5×10^{-4}. A total of 30 epochs were trained. A joint training strategy is employed for the loss function, whereby the pre-trained model is initially obtained through Softmax, and then further optimized through ArcFace to enhance the intra-class interval.

B. ArcFace LOSS FUNCTION HYPERPARAMETER SETTING EXPERIMENT

Firstly, VarKFaceNet1 and VarKFaceNet2 were trained using the Softmax function with a batch size of 128. The total number of training iterations was 0.6M. The training loss is illustrated in Fig. 6. The results of the accuracy test on the LFW dataset are presented in Fig. 7. From the figure, it can be seen that VarKFaceNet1 and VarKFaceNet2 have a high degree of model representation and achieve excellent performance on the LFW test set at an early stage of the model's development. VarKFaceNet1 is more readily convergent than VarKFaceNet2 and demonstrates superior performance.

Secondly, the impact of ArcFace hyperparameters s and m on the model based on Softmax pre-training was investigated. The accuracy of the LFW test set is shown in Table 4. It was noted that the model performance exhibited a dependency on the value of s, while the value of m remained constant. The optimal value for s was considered to be 16. Consequently, we selected 32 as the optimal value for s. When m was varied, the model exhibited slightly enhanced performance at $m = 0.15$ relative to other cases. This is due to the fact that for VarKFaceNet, an angular margin that is excessively large will render the network training process exceedingly challenging, whereas an angular margin that is excessively small will result in an increase in intra-class variability, thereby negatively impacting the model's performance. Consequently, the optimal hyperparameters for ArcFace were identified as $s = 32$ and $m = 0.15$.

C. COMPARISON EXPERIMENT OF LIGHTWEIGHT FACE RECOGNITION ALGORITHMS

First, we compare the visualized feature maps of VarKFaceNet1 with the large-scale EfficientNet and the

TABLE 4. Accuracy with different *s* and *m* values.

s	m	Accuracy (%)	s	m	Accuracy (%)
16	0.25	99.43	32	0.10	99.48
25	0.25	99.48	32	0.15	99.50
32	0.25	99.52	32	0.20	99.51
64	0.25	99.49	32	0.25	99.50
75	0.25	99.48	32	0.30	99.48
80	0.25	99.45	32	0.35	99.48

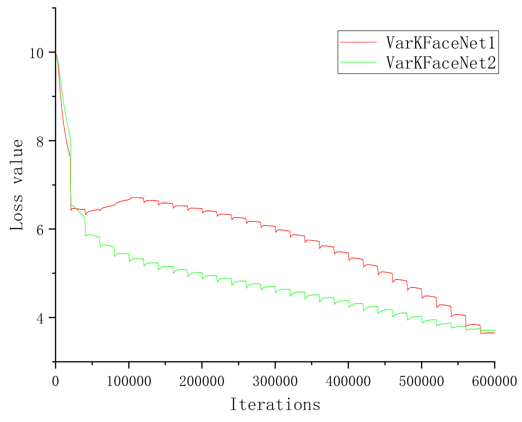

FIGURE 6. Training loss comparison of VarKFaceNet1 and VarKFaceNet2. This graph depicts the loss curves for VarKFaceNet1 and VarKFaceNet2 over 600,000 iterations. Both models exhibit a sharp decline in loss initially, with VarKFaceNet1 stabilizing at a higher loss value compared to VarKFaceNet2, suggesting differences in learning dynamics and efficiency between the two architectures.

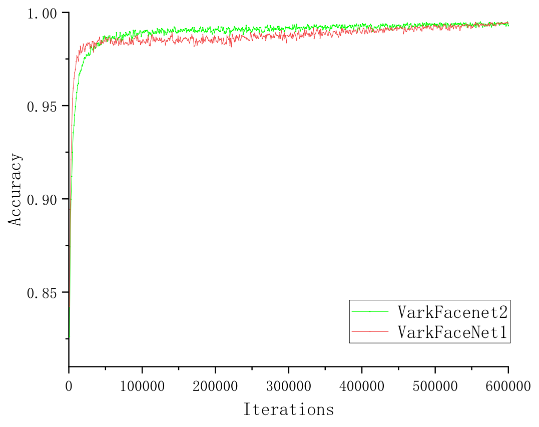

FIGURE 7. Accuracy progression on LFW for VarKFaceNet1 and VarKFaceNet2. The accuracy curves over 600,000 iterations demonstrate the performance of VarKFaceNet1 and VarKFaceNet2 on the Labeled Faces in the Wild (LFW) dataset. Both models achieve near-perfect accuracy, with VarKFaceNet2 showing slightly higher and more stable accuracy earlier in training, highlighting its effectiveness in handling real-world variability.

lightweight GhostFaceNet [32] in the mid and late stages. The results are shown in Figure 8. It can be observed that EfficientNet is able to extract more detailed information, and the overall feature map is more delicate. In contrast, GhostFaceNet, due to its smaller number of parameters, extracts more shallow features but preserves the overall features of the face. The feature extraction results of VarKFaceNet1 are generally similar to those of GhostFaceNet, but its extracted features are more abstract and favor contour information. This is mainly because VarKFaceNet1 introduces partial convolution of medium-large kernels, resulting in more shape preference.

Second, we compare the test results of popular convolutional neural networks in recent years, including ResNet50, EfficientNet, MobileNetV2, MobileFaceNet, MobileFaceNetV1, ShuffleFaceNet, VarGFaceNet, and Ghostfacenet on various popular datasets. The experimental results for comparing the performance of the models are shown in Table 3.

The performance of VarKFaceNet is somewhat degraded compared to both complex networks and the lightweight networks that have become popular in recent years, especially on datasets that include pose variations. Specifically, VarKFaceNet's performance on the LFW dataset is essentially the same as that of the best-performing lightweight model, MobileFaceNet. However, its performance on the CPLFW, CFP_FF, CFP_FP, VGG2_FP, and CALFW datasets decreases by 1.1%, 0.31%, 2.53%, 0.65%, and 0.31%, respectively. The performance degradation of VarKFaceNet is not significant on the CFP_FF and CALFW datasets. However, in the tests on the CFP_FP dataset, VarKFaceNet's performance decreased by 2.53%, likely because CFP_FP contains a large number of frontal and side images. This diversity of viewpoints requires face recognition algorithms to have a stronger ability to express texture details, which is currently lacking in VarKFaceNet.

Nevertheless, VarKFaceNet has fewer parameters and requires less computation than the lightweight models prevalent in recent years. Specifically, its computation is 30% less than that of MobileFaceNet, and the number of parameters is halved. The inference speed of VarKFaceNet is even more advantageous because it turns into a single-path structure during inference. Compared to mainstream models, VarKFaceNet has more obvious advantages in terms of speed and efficiency.

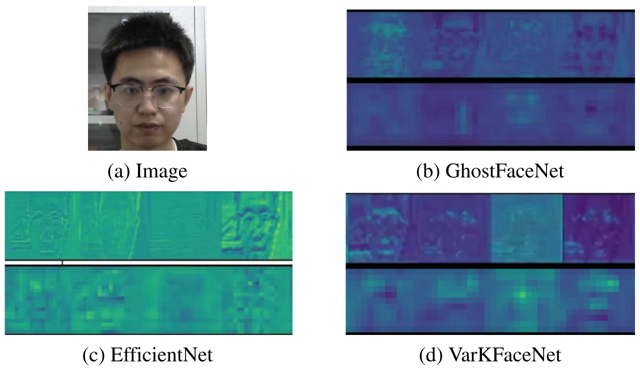

FIGURE 8. Comparative feature maps of different networks. (a) Original input image. (b) GhostFaceNet captures shallow feature layers preserving essential facial features due to fewer parameters. (c) EfficientNet extracts detailed and delicate feature maps, reflecting its capacity for deeper and more complex feature extraction. (d) VarKFaceNet's feature maps are more abstract, focusing on contour information, influenced by its use of medium-large convolutional kernels which enhances shape recognition.

TABLE 5. Performance comparison of various models.

Method	LFW	CPLFW	CFP_FF	CFP_FP	VGG2_FP	CALFW	PARAM	FLOPs
ResNet-50	99.64%	90.57%	99.63%	94.94%	92.84%	95.28%	40.29M	2.19G
EfficientNet	99.53%	90.92%	99.50%	96.32%	94.32%	95.78%	6.58M	1.14G
MobileNet-V2	99.55%	89.43%	99.48%	93.17%	91.58%	95.34%	2.26M	0.43G
MobileFaceNet	99.53%	90.34%	99.55%	95.17%	92.49%	95.42%	1.0M	0.45G
ShuffleFaceNet	99.70%	88.50%	99.60%	96.30%	-	95.05%	2.60M	0.58G
VarGFacenet	99.70%	88.55%	99.50%	96.90%	-	95.15%	5M	1.02G
Ghostfacenet	99.48%	89.13%	99.09%	92.74%	92.30%	94.53%	0.85M	0.15G
varkfacenet1	99.52%	89.23%	99.24%	92.64%	91.84%	95.11%	0.7M	0.24G
varkfacenet2	99.50%	88.13%	99.14%	92.24%	91.32%	94.8%	0.68M	0.21G

D. PERFORMANCE TESTING EXPERIMENT ON IJBB/C DATASETS

The IJB-B dataset contains 11,754 face images, 55,026 video frames, 7,011 videos, and 10,044 non-face images for 1,845 subjects. The IJB-C dataset contains 21,294 face images and 10,040 non-face images for 3,531 subjects. We present the receiver operating characteristic (ROC) curves of VarKFaceNet1 on the IJB-B and IJB-C evaluation sets in Figures 9 and 10. When the false positive rate (FPR) is 1E-5, the true positive rate (TPR) is close to 75% on IJB-B and 0.83 on IJB-C. The area under the curve (AUC) for VarKFaceNet1 is 99.5233% on IJB-C and 99.4041% on IJB-B, indicating slightly lower performance on IJB-B. Overall, VarKFaceNet1 demonstrates successful performance on the IJB-B and IJB-C datasets for large-scale face recognition.

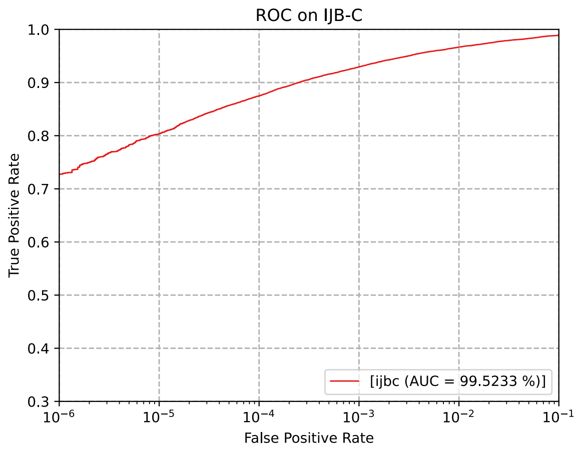

FIGURE 9. ROC Curve on IJB-C. The ROC curve demonstrates VarKFaceNet's performance on the IJB-C dataset, highlighting an AUC of 99.5233%. The curve illustrates the model's ability to maintain a high true positive rate (TPR) even at very low false positive rates (FPR), affirming its robustness in face recognition under challenging scenarios.

We also provide comparative results with the large-scale model ResNet-50, the best-performing lightweight MobileFaceNet, and the smallest volume GhostFaceNet on the IJB-B and IJB-C evaluation sets, as shown in Figures 11 and 12. From the figures, we observe that our model lags behind ResNet-50 and MobileFaceNet. This is due to the narrower width of our model compared to ResNet-50 and MobileFaceNet, resulting in fewer texture features being captured and decreased recognition accuracy in complex scenes. However, the performance of VarKFaceNet1 is comparable to GhostFaceNet, which has significantly fewer parameters. Although GhostFaceNet has fewer parameters, VarKFaceNet1's smaller size and single-path inference structure provide a speed advantage.

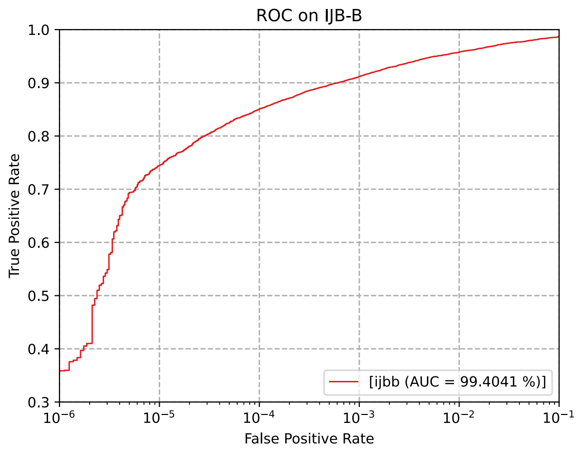

FIGURE 10. ROC Curve on IJB-B. This figure presents the ROC curve for the IJB-B dataset, where VarKFaceNet achieves an AUC of 99.4041%. The graph underscores the model's effectiveness in handling diverse and complex facial recognition environments, albeit slightly lower than its performance on IJB-C.

TABLE 6. Jetson Nano hardware configuration.

Server Hardware	Hardware Configuration
CPU	Arm Cortex-A78AE
GPU	NVIDIA Ampere(4GB)
Memory	8GB
Hard Disk Drive	128GB
Operating System	Linux Ubuntu 20.04.6 LTS

E. HARDWARE SYSTEM TESTING EXPERIMENT

Our test equipment, the NVIDIA Jetson Nano platform, is configured as shown in Table 6. An inference speed test and a real-scene recognition effect test were performed. The experimental results are presented in Table 7. When we converted the VarKFaceNet model into a model suitable for the Jetson platform, we did not perform any model

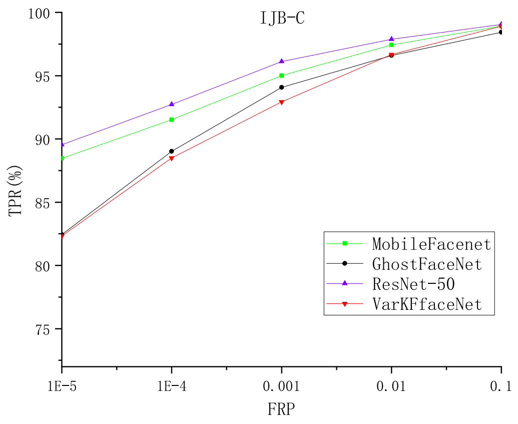

FIGURE 11. Comparative ROC curves on IJB-B. Showcasing ROC curves for VarKFaceNet alongside large-volume models like ResNet-50 and lightweight leaders like MobileFaceNet. VarKFaceNet, with fewer parameters, closely matches the performance of more resource-intensive models, demonstrating effective feature extraction capabilities despite its smaller size.

TABLE 7. Inference performance comparison.

Method	Actual Inference Time	PARAM	FLOPS
ResNet50	2.1s	40.29 M	2.19G
EfficientNet	55.7ms	6.58M	1.14G
MobileNetV2	29.4ms	2.26M	0.43G
MobileFaceNet	31.3ms	1.0M	0.45G
GhostFaceNet	17.6ms	0.85M	0.15G
VarKFaceNet1	13.2ms	0.7M	0.24G
VarKFaceNet2	12.5ms	0.68M	0.21G

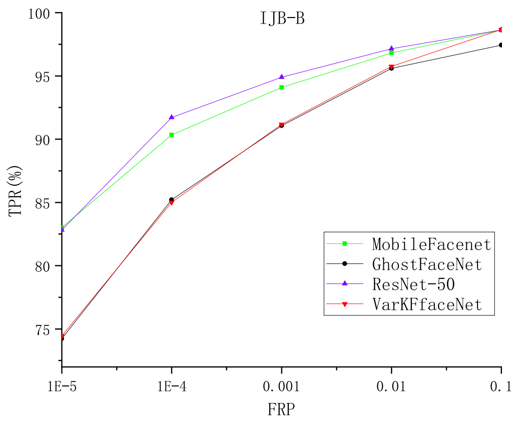

FIGURE 12. Comparative ROC curves on IJB-C. This graph compares VarKFaceNet's ROC performance with that of ResNet-50 and MobileFaceNet on the IJB-C dataset, revealing a competitive edge in terms of speed due to its streamlined one-way structure, while maintaining comparable accuracy to GhostFaceNet.

quantization operations, and the model size did not change. Therefore, the model accuracy performance remained almost unchanged. Experiments show that the recognition speed of VarKFaceNet1 is 13ms, which is 159 times, 4.2 times, 2.4 times, 2.2 times, and 1.3 times faster than ResNet-50, EfficientNet, MobileFaceNet, MobileNetV2, and GhostFaceNet, respectively. Therefore, VarKFaceNet1 outperforms other lightweight models in embedded devices in terms of speed and is suitable for deployment in embedded devices with minimal resource requirements. Meanwhile, its performance degradation is not significant, thus satisfying the accuracy requirements of face recognition technology.

V. CONCLUSION

Inspired by RepLKNet and VarGNet, we propose an efficient and lightweight network structure called VarKNet. VarKNet is designed with three branches: the backbone branch, the feature extraction branch, and the shortcut branch. VarKNet dynamically adjusts the convolution kernel size of the backbone branch according to the network width, effectively addressing the imbalance between deep convolution and pointwise convolution computations in embedded devices. VarKNet employs convolution kernels of various scales, providing a combined receptive field. Additionally, the introduction of medium and large kernel convolutions enhances both feature extraction capability and efficiency. Consequently, the need to expand too many channels to compensate for accuracy loss due to depthwise separable convolutions is significantly reduced, making VarKNet friendly for embedded devices.

Furthermore, VarKNet can transform its three-branch structure during training into a single-path structure through structural re-parameterization, effectively improving inference speed on embedded devices.

VarKFaceNet is built on VarKNet, achieving an accuracy of 99.5% on the LFW dataset and an inference speed of only 13.2ms on a Jetson Nano, striking a good balance between speed and accuracy. It is worth mentioning that this is the first time a non-traditional small kernel face recognition expert network has shown exceptional performance on embedded devices. However, there is still room for improvement in VarKFaceNet. Since face recognition tasks should have more texture preference, introducing large kernel convolutions too early may cause feature contamination. Therefore, VarKFaceNet requires more refined structural design. Additionally, attention mechanisms are an effective means to improve recognition performance, which the authors have not used due to concerns about excessive computational load. VarKFaceNet also needs to explore the most suitable loss function to improve model accuracy. Furthermore, by introducing convolution kernels of different scales, it is believed that VarKNet should exhibit more shape preference, which is very beneficial for detection tasks involving targets of different scales. We are currently exploring the application of VarKNet in object detection, and related work will be published soon.

REFERENCES

[1] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 15979–15988.

[2] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6655–6659.

[3] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[4] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.
[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[6] F. Zhao, P. Zhang, R. Zhang, and M. Li, "DGFaceNet: Lightweight and efficient face recognition," *Eng. Appl. Artif. Intell.*, vol. 124, Sep. 2023, Art. no. 106513.
[7] 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*.
[8] 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*.
[9] 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.
[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[11] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Scaling up your kernels to 31×31: Revisiting large kernel design in CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 11963–11975.
[12] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," 2014, *arXiv:1404.1869*.
[13] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002.
[14] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style ConvNets great again," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 13733–13742.
[15] Q. Zhang, J. Li, M. Yao, L. Song, H. Zhou, Z. Li, W. Meng, X. Zhang, and G. Wang, "VarGNet: Variable group convolutional neural network for efficient embedded computing," 2019, *arXiv:1907.05653*.
[16] 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.
[17] 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.
[18] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.
[19] Y. Tang, K. Han, J. Guo, C. Xu, C. Xu, and Y. Wang, "GhostNetv2: Enhance cheap operation with long-range attention," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 9969–9982.
[20] J. Chen, S. H. Kao, H. He, W. Zhuo, S. Wen, C. H. Lee, and S. H. G. Chan, "Run, don't walk: Chasing higher FLOPS for faster neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 12021–12031.
[21] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices," in *Biometric Recognition: 13th Chinese Conference, CCBR 2018, Urumqi, China, August 11–12, 2018, Proceedings 13*. Springer, 2018, pp. 428–438.
[22] Y. Martindez-Díaz, L. S. Luevano, H. Mendez-Vazquez, M. Nicolas-Diaz, L. Chang, and M. Gonzalez-Mendoza, "ShuffleFaceNet: A lightweight face architecture for efficient and highly-accurate face recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2721–2728.
[23] C. N. Duong, K. G. Quach, I. Jalata, N. Le, and K. Luu, "MobiFace: A lightweight deep learning face recognition on mobile devices," in *Proc. IEEE 10th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Sep. 2019, pp. 1–6.
[24] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2990–2999.
[25] X. Zeng, Q. Dong, and Y. Li, "MG-CNFNet: A multiple grained channel normalized fusion networks for medical image deblurring," *Biomed. Signal Process. Control*, vol. 82, Apr. 2023, Art. no. 104572.
[26] Z. Bai, J. Wang, X.-L. Zhang, and J. Chen, "End-to-end speaker verification via curriculum bipartite ranking weighted binary cross-entropy," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 30, pp. 1330–1344, 2022.
[27] Z. Wang, B. Huang, G. Wang, P. Yi, and K. Jiang, "Masked face recognition dataset and application," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 5, no. 2, pp. 298–304, Apr. 2023.
[28] C. Lu and X. Tang, "Surpassing human-level face verification performance on LFW with GaussianFace," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2015, vol. 29, no. 1, pp. 1–9.
[29] M. Tramier, M. Zahid, J. Mevel, M. Masse, and M. Coppey-Moisan, "Sensitivity of CFP/YFP and GFP/mCherry pairs to donor photobleaching on FRET determination by fluorescence lifetime imaging microscopy in living cells," *Microsc. Res. Technique*, vol. 69, no. 11, pp. 933–939, Nov. 2006.
[30] Q. Wang, T. Wu, H. Zheng, and G. Guo, "Hierarchical pyramid diverse attention networks for face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8323–8332.
[31] F. Zhao, P. Zhang, R. Zhang, and M. Li, "UnifiedFace: A uniform margin loss function for face recognition," *Appl. Sci.*, vol. 13, no. 4, p. 2350, Feb. 2023.
[32] F. Zhao, P. Zhang, and R. Zhang, "Research on lightweight face recognition algorithm based on GhostNet," *Electron. Meas. Technol.*, vol. 45, no. 16, pp. 130–136, 2024.

QINGHUA MA (Graduate Student Member, IEEE) received the bachelor's degree in engineering from Lanzhou Jiaotong University, in 2019. He is currently pursuing the master's degree with North University of China. He has developed several face recognition attendance prototypes based on the Jetson platform. His research interests include multi-target detection and recognition, intelligent perception, and multi-sensor fusion.

PENG ZHANG received the Ph.D. degree in mechanical design and manufacturing and automation from Lanzhou University of Technology. He is currently working as an Associate Professor with the School of Instrumentation and Electronics, North University of China. He has published more than ten papers. His current research interests include inertial devices and micro-inertial combined navigation, inertial sensing microsystems, and intelligent sensing systems for multiple sensors.

MIN CUI received the Ph.D. degree. She is currently working as an Associate Professor. She is also the Director of China Computerized Automatic Measurement and Control Technology Association. She has authorized five invention patents and published more than ten SCI and other high-level papers. Her research interests include automated test and control, intelligent perception, and control.

• • •