JEI Journal of Electronic Imaging

RESEARCH PAPER

LW-DCGAN: a lightweight deep convolutional generative adversarial network for enhancing occluded face recognition

Yingying Lv, Jianping Wang,* Guohong Gao, and Qian Li
Henan Institute of Science and Technology, School of Computer Science and Technology, Xinxiang, China

ABSTRACT. Traditional facial recognition techniques often struggle to balance accuracy with model complexity. High accuracy typically demands intricate models, slowing recognition speeds on devices such as smartphones. Conversely, faster methods often sacrifice accuracy. We introduce a lightweight deep convolutional generative adversarial network (LW-DCGAN), designed specifically to address the challenges of occluded face recognition. By simplifying the network architecture and employing efficient feature extraction techniques such as transpose convolution, batch normalization, feature pyramid networks, and attention modules, we enhance both hierarchical sampling and contextual relevance. Furthermore, *L*1 regularization and channel sparsity techniques compress the model for resource-constrained environments. We thoroughly evaluate LW-DCGAN's generalization and robustness, comparing its performance against other generative adversarial network variants and common face recognition models. The results demonstrate that LW-DCGAN achieves higher accuracy while significantly reducing model size and computational overhead, offering a promising advancement in lightweight face recognition technology.

Keywords: deep learning; occluded face recognition; generative adversarial network; deep convolutional generative adversarial network

Paper 240365G received Apr. 13, 2024; revised Sep. 11, 2024; accepted Sep. 12, 2024; published Oct. 25, 2024.

1 Introduction

With the advancement of artificial intelligence and computer vision, face recognition technology has become widely applicable across various fields. Addressing the challenge of efficiently and accurately recognizing face occlusions—caused by factors such as masks and glasses—is crucial due to its impact on feature extraction, data acquisition, and computational complexity. The generative adversarial network (GAN), a deep learning model featuring a generator and a discriminator, offers a solution by fostering an iterative learning process among these components. This method aims to enhance the generator's output to more closely resemble real data distributions.[1] The deep convolutional generative adversarial network (DCGAN) advances this concept by replacing fully connected layers with convolutional ones, thereby stabilizing and simplifying the network architecture for quicker convergence.[2] This adaptation significantly improves the feasibility of accurately and efficiently addressing occluded face recognition challenges.

Specifically, DCGAN has played a pivotal role in enhancing generative capabilities for occluded face recognition, allowing for the creation of high-quality images. This capability is critical for developing more sophisticated recognition models. In addition, it effectively learns

*Address all correspondence to Jianping Wang, wangjianping@hist.edu.cn

to capture the nuances of occlusion patterns, enhancing the realism of generated images and refining the discriminator's ability to detect these features.[3] Moreover, DCGAN demonstrates effectiveness in mitigating common occlusions, such as masks and glasses, thereby preserving facial feature integrity.[4]

This paper introduces a lightweight DCGAN (LW-DCGAN) specifically optimized for the challenges of occluded face recognition, with a focus on efficiency and accuracy suitable for mobile devices. The main contributions are as follows:

(a) We provide a comprehensive survey of lightweight occluded face recognition techniques and the latest advancements in adversarial generative networks, setting the stage for our LW-DCGAN approach.

(b) We introduce the LW-DCGAN model, featuring innovative lightweight generators and discriminators. This model excels in feature learning and image generation for occluded face recognition, offering a viable solution for real-world applications.

(c) We describe the LW-DCGAN architecture, including the design principles of generators and discriminators, model compression techniques, optimization strategies for loss functions, and training approaches tailored to limited computational resources.

(d) We outline a multi-level experimental strategy that begins with an ablation study to validate the components of LW-DCGAN, followed by an analysis of the training process and testing of LW-DCGAN's performance across multiple datasets.

(e) Finally, we conduct a comparative evaluation of LW-DCGAN against GAN variants and several face recognition models, focusing on occluded face recognition accuracy, efficiency, and aspects such as model size and inference time.[5,6]

The paper is outlined as follows: Section 2 reviews the related research. Section 3 presents the LW-DCGAN architecture and algorithmic details. Section 4 explains the experimental setups and analyses. Section 5 discusses the implications of our findings. Section 6 concludes the paper.

2 Related Work

2.1 Advances in Occluded Face Recognition Algorithms

Recent advancements in occluded face recognition have significantly improved the robustness and accuracy of recognition algorithms under challenging conditions. The sparse representation classification (SRC) method, introduced by Wright et al.,[7] became a foundational approach for occluded face recognition. By employing $L1$ norm regression, SRC derived sparse coefficients, which proved effective in classifying occluded faces. It attained recognition rates of 98.1% on the Extended Yale dataset and 94.1% on the augmented reality (AR) face database. In 2011, Zhang et al.[8] proposed the collaborative representation method (CRC), which utilized least squares for coefficient estimation, thereby reducing computational complexity. CRC demonstrated recognition rates exceeding 90% on various experimental datasets. Ou et al.[9] introduced a sparse representation-based classification (SSRC) method that incorporated an occlusion dictionary. SSRC achieved recognition rates of 97.2% on the Extended Yale B and 95.3% on the AR datasets for non-occluded faces, whereas for occluded faces, the rates were 95.8% and 92.7%, respectively. Zheng et al.[10] devised an iterative robust coding technique using a Laplacian–Uniform hybrid approach, which ensured high recognition accuracy even under challenging conditions such as occlusion and pixel damage. Despite these improvements, traditional methods remained limited to "shallow" facial features, often missing finer details.

In recent years, the rapid advancement of deep learning has ushered in significant progress in occluded face recognition. Mundial et al.[11] leveraged supervised learning techniques to identify masked faces, achieving an accuracy rate of up to 97% through facial features extracted by deep neural networks. Vu et al.[12] combined deep learning with local binary pattern features to capture information from the eyes, forehead, and eye sockets of masked faces. These features were then merged with those learned from a face detector, creating a unified framework for masked face recognition. Their system recorded an $F1$-score of 87% on the COMASK20 dataset and 98% on the Essex dataset.

Montero et al.[13] engineered an end-to-end face recognition model based on the ArcFace architecture, incorporating data enhancement and dynamic dataset mixing. This approach resulted in an average accuracy of 98% in recognizing masked faces. Hariri[14] proposed an efficient method tailored for recognizing masked faces during the coronavirus disease 2019 (COVID-19) pandemic. The approach employed VGG-16, AlexNet, and ResNet-50 for feature extraction and quantization, followed by a multi-layer perceptron (MLP) classifier. On the real-world masked face recognition dataset (RMFRD), VGG-16 achieved a recognition rate of 91.3%, ResNet-50 reached 89.5%, and AlexNet secured 86.6%. Golwalkar and Mehendale[15] introduced FaceMaskNet-21, a neural network optimized for masked face recognition using multiple convolutional and fully connected layers. Validated on various datasets, the network demonstrated 88.92% accuracy with an execution time of under 10 ms. Zhang et al.[16] developed a lightweight occluded face recognition model based on MobileNetV2. They replaced the average pooling in the attention module with a depth-wise separable convolution and integrated an improved dual attention module. Their model achieved accuracies of 90% and 91% on the mask-labeled faces in the wild (LFW) and mask-AgeDB datasets, respectively. Huang et al.[17] proposed a progressive learning loss for face recognition (PLFace) method, implementing a progressive training strategy for deep face recognition. PLFace adaptively adjusted the weights of masked and unmasked samples at different training stages. Experiments revealed an average accuracy of 77% on the RMFRD dataset, 99.7% on the LFW dataset, and 94% on the IJB-C 1:1 validation. Ge et al.[18] introduced a convolutional visual self-attention network (CVSAN) that combined local convolutional features with self-attention for long-range dependencies. On the Masked VGGFace2 dataset, CVSAN surpassed ArcFace, increasing accuracy by 0.8% on LFW and boosting TPR@FAR = 0.1% from 89.90% to 95.16% on SM-LFW.

Cheng et al.[19] implemented a face recognition system by Google (FaceNet) with transfer learning, using InceptionResNetV2, InceptionV3, and MobileNetV2. They incorporated a cosine annealing mechanism, which enhanced accuracy to ~93% across all three models. In recent studies, Zhong et al.[20] introduced masked uncertainty fluctuation to measure sample identifiability by combining feature amplitude and variance uncertainty. The approach resulted in an average accuracy improvement ranging from 1.33% to 13.28%. Faruque et al.[21] designed a lightweight convolutional neural network model that integrated batch normalization, dropout, and depth normalization to optimize overall performance. Compared with other deep learning models, this model achieved a high recognition accuracy of 97%. Sharma et al.[22] developed a novel dual method for masked face detection using AntelopeV2, which utilized the RetinaFace detection algorithm and the ResNet100 convolutional neural network for face detection and embedding generation. Experimental results indicated an accuracy of ~98.5%. The research outlined above explored diverse approaches to occluded face recognition, addressing challenges through various models, data augmentation techniques, and feature fusion methods. However, several limitations and challenges persisted:

(a) Although many studies demonstrated promising results in controlled environments or specific datasets, the generalization of occluded face recognition algorithms to diverse real-world scenarios remained limited. Adapting models to varying lighting conditions, facial orientations, and occlusion types posed a significant challenge.

(b) Existing algorithms often struggled to handle complex occlusion scenarios, such as partial occlusions, overlapping occlusions, or occlusions caused by accessories such as sunglasses or hats. Developing robust algorithms capable of accurately recognizing faces under diverse occlusion conditions was essential for practical applications.

(c) Despite the progress in algorithm development, many existing methods required substantial resources, limiting their practical deployment in resource-constrained environments. Enhancing the computational efficiency of occluded face recognition algorithms while maintaining high accuracy remained a critical research area.

2.2 Progress on GANs for Facial Recognition

The application of GANs in facial recognition tasks has gained significant attention, particularly in addressing challenges such as occlusion and low-quality images. Li et al.[23] introduced a masked face recognition method using deocclusion distillation, which combined GAN and

attention mechanisms to predict and reconstruct facial features. Experiments demonstrated that this approach improved accuracy by 1.3% over VGGFace and 0.2% over VGGFace2. Fu et al.[24] presented a GAN-based unsupervised low-light image enhancement network with an attention module to improve image quality. Using an autoencoder, the method adapted enhancement across regions, highlighting details and reducing noise. It achieved a peak signal-to-noise ratio (PSNR) of 21.523 and a structural similarity index measure (SSIM) of 0.812 on the paired normal/lowlight images (PNLI) and low-light (LOL) test sets.

Chen et al.[25] enhanced face detection with an improved Xception model incorporating a local GAN. By replacing standard convolutions with inception blocks using dilated convolutions, the model effectively captured multi-scale features and achieved over 90% accuracy in detecting small-area faces. Zhang et al.[26] developed a domain embedding GAN for face repair, integrating three types of face domain knowledge into a hierarchical variational autoencoder to guide the repair process. Experiments showed that domain embedded generative adversarial network (DE-GAN) surpassed leading image inpainting methods on CelebA and CelebA-HQ datasets, achieving SSIM scores of 0.893 and 0.895 and PSNR scores of 26.132 and 26.208, respectively. Lin et al.[27] proposed a face de-identification method using GAN with a seven-layer network and two discriminators to boost feature extraction. Their model, evaluated through pixel loss, content loss, and adversarial loss, achieved over 90% recognition accuracy across various datasets.

Zhang et al.[28] introduced a GAN-based method that utilized contextual information to detect small-sized faces in complex environments. They generated virtual images with rich contextual information using GAN, fused these with real images, and created a comprehensive dataset for training deep learning models. Trigueros et al.[29] devised a method for generating realistic training data using GANs. The approach combined synthetic images with real images and employed a multi-scale generator network architecture to capture more details and variations. Experiments on the wider facial detection in the wild (WIDER FACE) and face detection data set and benchmark (FDDB) datasets demonstrated the effectiveness of their method in recovering clear, high-resolution faces from small, blurry ones. Yang et al.[30] constructed a semantic face restoration method using a dual discriminator DCGAN. Leveraging the VGG16 network to learn deep image features, their model achieved clearer and more realistic restoration results at the pixel level. Experiments on the CelebA dataset reported a PSNR of over 26 on most test datasets.

Hong et al.[31] proposed a two-stage face inpainting method. The first stage predicted facial landmarks to provide geometric and symmetry information for the GAN. In the second stage, the masked face image and corresponding facial feature points were input into a GAN to inpaint the missing areas. In experiments, their method achieved SSIM and PSNR scores of 0.9 and above 30, respectively, outperforming the light adaptive face image normalization. Huang et al.[32] introduced Cycle Style GAN, which integrated the pre-trained Style-GAN 3 network into the Cycle-GAN architecture for near-infrared to visible (NIR-VIS) cross-domain learning. This model synthesized realistic visible images from near-infrared (NIR) images and achieved a rank 1 accuracy of 99.6% on the CASIA NIR-VIS 2.0 database. The research mentioned above illustrates various contributions of adversarial networks in the field of face recognition. However, they still face certain limitations and challenges:

(a) Despite significant efforts to enhance performance, there remains the lack of focus on simplifying and compressing models, which limits their practical applicability, especially in resource-constrained environments, thereby hindering efficient deployment.

(b) Currently, the application of adversarial networks in face recognition is primarily centered around image restoration, generation, and enhancement. However, there are relatively few studies that directly use adversarial networks for recognition models, indicating significant potential and ample research opportunities in this field warranting further exploration and development.

2.3 Studies on Lightweight Network Architecture

The development of lightweight network architectures has become increasingly crucial for deploying deep learning models in resource-constrained environments, such as mobile devices. These architectures aim to reduce computational complexity and memory usage while maintaining high accuracy, particularly in tasks such as facial recognition where model efficiency is critical. Andrew and Menglong[33] introduced the MobileNet model, which employed depth-wise separable

convolution to reduce parameters and computational complexity. In face attribute classification, MobileNet achieved 88.7% mean average precision (mAP) with just 1% of the computation. It recorded a mAP of 19.3% in common objects in context (COCO) object detection and maintained 79.4% accuracy in face embedding tasks, all while significantly reducing model parameters.

Sandler et al.[34] presented MobileNetV2, featuring an inverted residual architecture and linear bottlenecks, which improved efficiency and representation. Compared with MobileNetV1, MobileNetV2 achieved the same 22.1% mAP on COCO object detection while cutting parameters by 16%, computational load by 38%, and runtime by 26%. Howard et al.[35] further advanced this by proposing MobileNetV3, which integrated automated machine learning techniques to optimize its lightweight design. By employing network architecture search technology, the model automatically identified the optimal network architecture and introduced the efficient hard Swish activation function, further reducing computational overhead. Replacing SSDLite's feature extractor with MobileNetV3 yielded a 27% speed improvement over MobileNetV2. Zhang et al.[36] developed ShuffleNet, which used group convolution and channel shuffle to lower computational complexity and enhance information flow. On the ImageNet 2012 dataset, experiments showed that ShuffleNet reduced classification error by 3.1% and computational complexity by 45 MFLOPs compared with MobileNet.

Ma et al.[37] introduced ShuffleNetV2, which enhanced group convolution and optimized feature transfer mechanisms. By simplifying network design and improving feature transfer efficiency, ShuffleNetV2 achieved better computational and storage efficiency on practical hardware, making it well-suited for low-power environments. At 500 MFLOPs, ShuffleNetV2 was 58% faster than MobileNetV2, 63% faster than ShuffleNetV1, and 25% faster than Xception. Tan and Le[38] presented EfficientNet, which employed compound scaling to optimize network width, depth, and resolution. This method ensured high efficiency across various resource constraints. On the ImageNet dataset, EfficientNet achieved a top accuracy of 97.1% with 66M parameters, surpassing MobileNetV2. Han et al.[39] introduced GhostNet, a model that created additional feature maps through linear transformations of existing ones, significantly lowering computational demands. Experiments showed that GhostNet outperformed other networks on the ImageNet dataset with a top 1 accuracy higher than MobileNetV3 by ∼0.5% while maintaining similar latency. Liu et al.[40] developed a lightweight convolutional neural network for real-time semantic segmentation. The network used branched skip connections to capture contextual information and applied factorized dilated depth-wise separable convolutions to learn features from various scales. Despite its small size of 0.8M parameters, the network processed 1024×512 images at 60 FPS on a single RTX 2080Ti graphics processing unit (GPU).

Chen et al.[41] investigated a parallel design that combined MobileNet and Transformer, leveraging the strengths of MobileNet in local feature processing and Transformer in global interactions. In ImageNet classification tasks, this design outperformed MobileNetV3 within the low FLOP range of 25M to 500M FLOPs. Lyu et al.[42] proposed a GPU-free real-time object detection method using a quantized single-shot multi-box detector (MobileNet-SSD), combining the lightweight design of MobileNet with the real-time detection of SSD. The quantized model significantly reduced computational and storage requirements. Experiments on a dataset of 22k monitoring images demonstrated a compression ratio of up to 21 times and a detection speed of nearly 25 FPS in a central processing unit (CPU)-only environment, with an mAP of 86.83% and a model size of 600 KB. Kavyashree and El-Sharkawy[43] enhanced the MobileNet baseline architecture and reduced its size to 2.3 MB through techniques such as weight quantization, model pruning, and channel pruning, achieving an accuracy of 89.13%.

Shi et al.[44] introduced DPNet, a dual-path network for efficient object detection with lightweight self-attention. It used a self-attention module in the backbone to encode global interactions and a multi-input version in the feature pyramid network (FPN) for cross-resolution interactions. On the COCO dataset, DPNet achieved 29.0% AP on the test-dev set with only 1.14 GFLOPs and a model size of 2.27M for 320×320 images. Jia et al.[45] designed a recognition model based on an improved YOLOv7 combined with the lightweight MobileNetV3. MobileNetV3 was used for feature extraction, reducing the number of parameters while integrating the coordinate attention (CA) mechanism and the SIoU loss function to enhance accuracy. The model was tested on image datasets, achieving an accuracy of 92.3%. The research into lightweight network architectures is extensive but several challenges remain:

(a) Despite the significant attention given to MobileNet and other established lightweight architectures, there has been a notable lack of research focused on developing specialized architectures for occluded face recognition.

(b) Currently, there exists no research that explores the integration of lightweight frameworks with GANs specifically for occluded face recognition.

To address this issue, studying the integration of lightweight network architectures with DCGANs could enhance the stability of network training. This combination would improve the model's ability to learn and interpret the complex features of occluded areas, thereby enhancing the accuracy of face recognition.

3 Methods

LW-DCGAN is a generative adversarial network specifically designed for addressing the challenge of occluded face recognition. It utilizes a streamlined convolutional network structure, enhanced with FPNs and attentional residual context modules (ARCM), to balance high-quality image generation with reduced model complexity. LW-DCGAN aims to improve the accuracy of recognizing occluded faces while maintaining a lightweight and efficient design, making it suitable for deployment in resource-constrained environments, such as on mobile devices for real-time applications.

3.1 Architecture of LW-DCGAN

The core architecture of LW-DCGAN features a generator and a discriminator. The generator utilizes a lightweight convolutional network architecture, integrating an FPN and an ARCM to progressively generate high-resolution images of occluded faces. Through the use of transposed convolution layers and Tanh activation, the generator upscales features to produce the final image output. The discriminator, on the other hand, extracts facial features through convolutional layers and incorporates a CA module to enhance feature representation. Furthermore, an auxiliary face recognition module is cascaded within the discriminator to support face recognition tasks. This design is intended to enhance feature extraction efficiency and image quality, thereby ensuring high-accuracy face recognition even in the presence of occlusion. The network architecture of LW-DCGAN is illustrated in Fig. 1.

3.2 Generator Network

3.2.1 *Lightweight convolutional module*

In the generator, the lightweight core component is the bottleneck block, which consists of a series of convolutional layers and activation functions. The first layer of the bottleneck block utilizes point-wise convolution to expand the number of channels, allowing for the extraction of more detailed feature information. The second layer employs depth-wise convolution to fuse

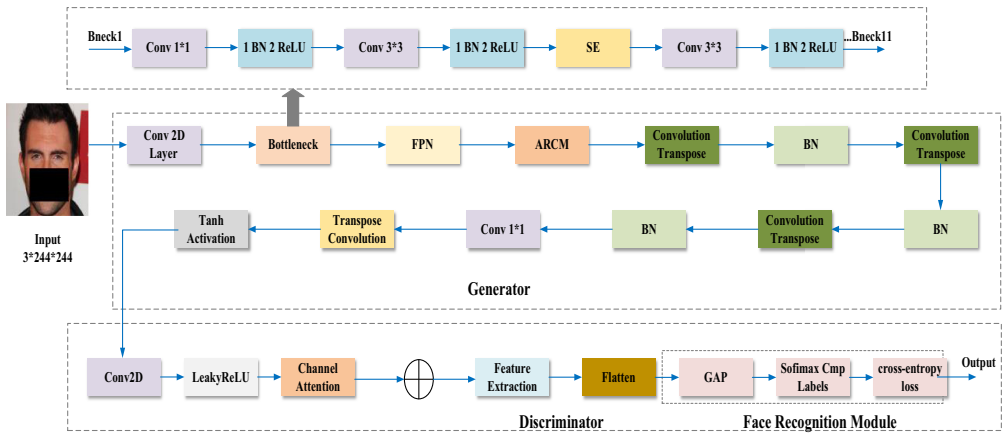


Fig. 1 Architecture of LW-DCGAN.

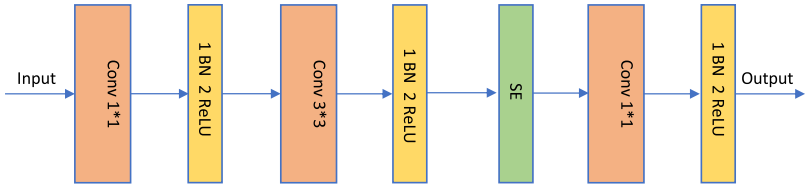


Fig. 2 Architecture of the bottleneck.

features, integrating information from different levels. The final layer again uses point-wise convolution, this time for dimensionality reduction, which decreases the computational load and the number of parameters. In addition, squeeze-and-excitation modules are incorporated into the network. These modules dynamically adjust the importance of channels by learning the relationship between effective and ineffective weights, thereby enhancing the network's expressive power and accuracy. Batch normalization (BN) is applied to stabilize and accelerate the training process, whereas rectified linear units (ReLU) are used as activation functions to introduce nonlinearity. The detailed architecture of the bottleneck block is illustrated in Fig. 2.

3.2.2 *FPN module*

The core of the FPN architecture is the establishment of top-down and bottom-up feature pathways. The top-down path is responsible for up-sampling high-level feature maps to match the size of the feature maps in the bottom-up path. Conversely, the bottom-up path connects low-resolution, high-level semantic features with high-resolution, low-level semantic features through feature fusion. This architecture enables shallow features to receive guidance from deeper features, thereby enhancing the detection capabilities of the shallow features. The entire process constructs a comprehensive feature pyramid, allowing features of different scales and semantic levels to be effectively utilized. In our model, we select layers 2, 3, 5, 9, and 12 from the feature extraction architecture to extract image features and merge them as multi-scale feature information within the FPN. The process of building the FPN network is illustrated in Fig. 3.

The input image undergoes a series of operations, including convolution and pooling, to form different scale feature layers C1, C2, C3, C4, and C5. These layers have an increasing down-sampling rate and decreasing resolution. The Input $\in R3 * 224 * 224$ with three channels and a size of 224×224 undergoes down-sampling, resulting in feature maps C1 to C5 with sizes $C1 \in R16 * 112 * 112$, $C2 \in R16 * 56 * 56$, $C3 \in R24 * 28 * 28$, $C4 \in R48 * 14 * 14$, and $C5 \in R96 * 7 * 7$. The convolution with a $1 * 1$ kernel to change the feature channel number to 64 and the subsequent $2\times$ up-sampling are applied, resulting in different-sized feature maps P5 $\sim$ P2. P5 is calculated from C5, and P4 is derived by combining P5 and C4, and so forth. The size is $P5 \in R64 * 7 * 7$, $P4 \in R64 * 14 * 14$, $P3 \in R64 * 28 * 28$, and $P2 \in R64 * 56 * 56$. The final output feature map set is $P = [P2, P3, P4, \text{and } P5]$.

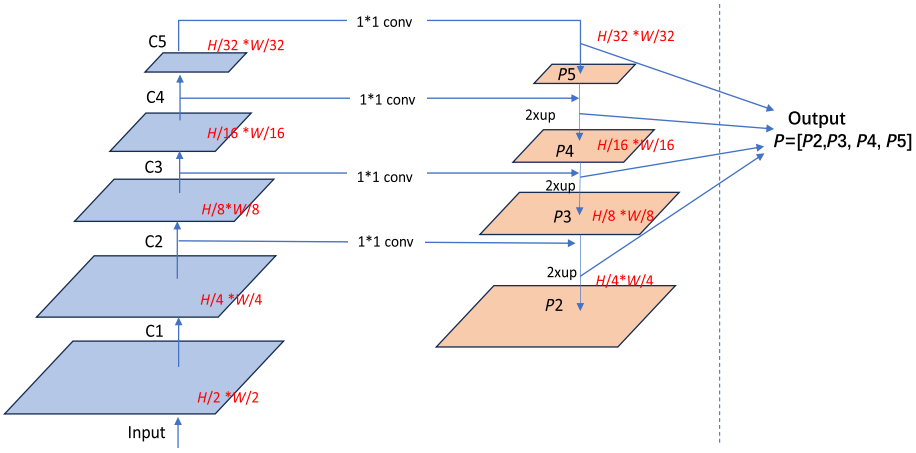


Fig. 3 FPN network architecture.

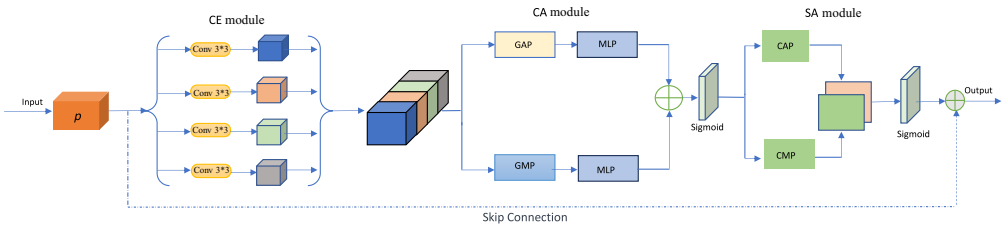


Fig. 4 ARCM module.

3.2.3 *ARCM module*

To enhance feature extraction in the unobscured regions, LW-DCGAN incorporates ARCM into the generator's design. This module boosts feature extraction capabilities, mitigates gradient vanishing issues, and addresses generator instability. The ARCM consists of three key components: a context enhancement (CE) module, a CA module, and a spatial attention (SA) module. The CE module employs four parallel branches, each utilizing 3×3 convolutional kernels, aimed at maintaining the same receptive field size while minimizing the overall number of parameters. In this context, the receptive field refers to the perceptual range of each neuron within the network in relation to the input data. Each branch generates a feature map, which are then concatenated to form an enriched context feature map. The concatenation operation is crucial in integrating diverse feature information from different branches, thereby enhancing the model's expressive capabilities. Following the cascading of the CE with the CA and SA modules, a skip connection is introduced to further stabilize the network. The structural framework of ARCM is depicted in Fig. 4.

The input to the ARCM is the output feature map collection from the FPN. Through deconvolution, the input feature maps are up-sampled by applying convolution, thereby unifying the size of all feature maps to a common resolution. These processed feature maps are then concatenated channel-wise, merging them into a single tensor. As the channel dimensions of P2, P3, P4, and P5 are already aligned, it is only necessary to concatenate them along the channel axis to form the tensor. This concatenation results in a context-enhanced feature map. The CA module then applies global average pooling (GAP) and global maximum pooling to the concatenated feature map, after which the results are passed through an MLP, which consists of a three-layer fully connected network. The final output is obtained through a normalized sigmoid function. The GAP operation is performed on the context-enhanced feature map to generate the channel attention feature map $\mathrm{CA}_{\mathrm{avg}}$, as illustrated in Eq. (1)

$$\mathrm{CA}_{\mathrm{avg}}(c,h) = \left(\frac{1}{W}\right)\sum_{i=0}^{W-1} CE(c,h,i), \tag{1}$$

where $\mathrm{CE}(c,h,i)$ denotes the value of the feature map at channel c, height h, and width i. The symbol W stands for the width of the feature map. Applying fully connected layers and a sigmoid activation function to $\mathrm{CA}_{\mathrm{avg}}$ results in the channel attention weight feature map $\mathrm{CA}_{\mathrm{weight}}$, as shown in Eq. (2)

$$\mathrm{CA}_{\mathrm{weight}} = \sigma(W_{ca} \times \mathrm{CA}_{\mathrm{avg}} + b_{ca}), \tag{2}$$

where W_{ca} means the weight of the fully connected layer, and b_{ca} is the bias of the fully connected layer. σ represents the sigmoid function. The fully connected layer receives $\mathrm{CA}_{\mathrm{avg}}$ as input and performs a linear transformation of weights and biases. The channel attention weight feature map $\mathrm{CA}_{\mathrm{weight}}$ and CE are multiplied element by element to obtain the channel attention enhancement feature map $\mathrm{CA}_{\mathrm{enhanced}}$. The workflow of the CA module is shown in Fig. 5.

The SA module first performs channel-wise average pooling and channel-wise max pooling, followed by a two-dimensional convolution. The resulting feature maps are then passed through a sigmoid function. Afterward, the feature maps are concatenated along the channel dimension. Finally, a skip connection is introduced to prevent information loss and mitigate gradient vanishing. The architecture of the spatial attention module is illustrated in Fig. 6.

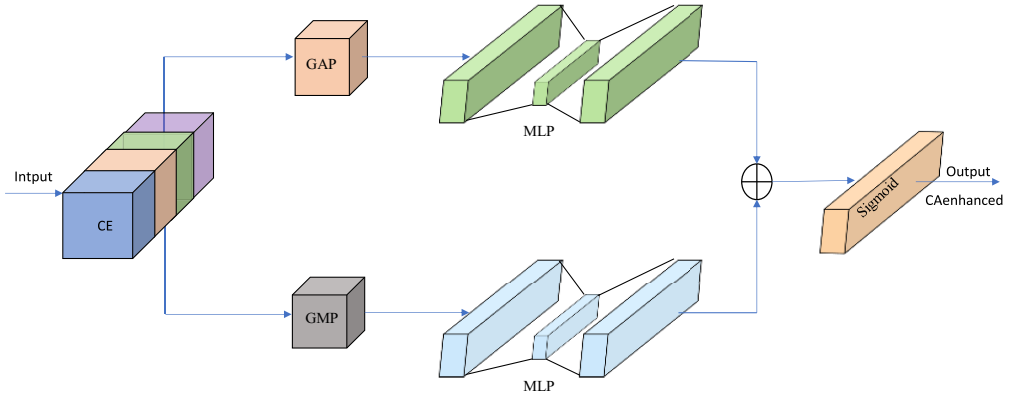


Fig. 5 CA module.

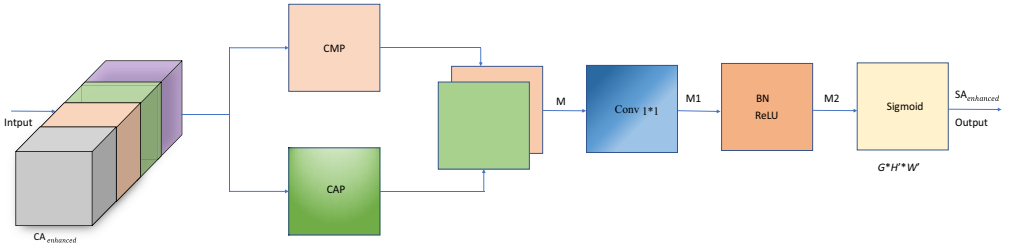


Fig. 6 SA module.

By applying both an average pooling kernel and a max pooling kernel to the channel attention-enhanced feature map $\mathrm{CA}_{\mathrm{enhanced}}$, we perform pooling operations in both horizontal and vertical directions. The feature map H_{pool} in the horizontal direction is obtained, as shown in Eq. (3)

$$H_{\mathrm{pool}}(c,h) = \left(\frac{1}{W}\right)\sum_{i=0}^{W-1}\mathrm{CA}_{\mathrm{enhanced}}(c,h,i) + \max_{i=0}^{W-1}\mathrm{CA}_{\mathrm{enhanced}}(c,h,i), \tag{3}$$

where $\mathrm{CA}_{\mathrm{enhanced}}(c,h,i)$ symbolizes the value of the i'th element in channel c and feature map h.

Simultaneously, in the vertical direction, we obtain the feature map W_{pool} by applying average pooling and max pooling as shown in Eq. (4)

$$W_{\mathrm{pool}}(c,w) = \left(\frac{1}{H}\right)\times\sum_{j=0}^{H-1}\mathrm{CA}_{\mathrm{enhanced}}(c,j,w) + \max_{j=0}^{H-1}\mathrm{CA}_{\mathrm{enhanced}}(c,j,w), \tag{4}$$

where H stands for the number of elements considered along the height direction during pooling. $\mathrm{CA}_{\mathrm{enhanced}}(c,j,w)$ denotes the value of the element in channel c and feature map at the w'th column and j'th row.

The width of the feature map W_{pool} is transposed to W, resulting in the transposed feature map $W_{\mathrm{pool\_transpose}}$. Concatenated with the feature map $W_{\mathrm{pool\_transpose}}$, the feature layer M is obtained. It is subject to a 1×1 convolution operation, resulting in the feature layer $M1$. Then, batch normalization and ReLU activation function are applied to $M1$, resulting in the feature layer $M2$. $M2$ is segmented, and the segmented parts are transposed and subject to a 1×1 convolution operation. The resulting feature maps are then passed through the sigmoid function to generate the final spatial weights. The spatial weights in the height direction are represented by $H'(i,k)$ as shown in Eq. (5), and the spatial weights in the width direction are represented by $W'(j,k)$ as shown in Eq. (6)

$$H'(i,k) = \sigma(H_{\mathrm{pool}}(i,k)), \tag{5}$$

$$W'(j,k) = \sigma(W_{\mathrm{pool}}(j,k)), \tag{6}$$

where i symbolizes the position coordinate along the height direction of the feature map, j represents the position coordinate along the width direction to the feature map, and k stands for the pixel value of a channel in the feature map.

Ultimately, the output of the ARCM module is $\mathrm{SA}_{\mathrm{enhanced}}$, as shown in Eq. (7)

$$\mathrm{SA}_{\mathrm{enhanced}} = \sum_{k=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} G(i,j,k) \times H'(i,k) \times W'(j,k), \tag{7}$$

where $G(i,j,k)$ denotes the value at position i in the original feature map.

3.2.4 *Network slimming*

To compress the generator network model into a more compact size for efficient deployment in resource-limited environments, such as mobile devices, our goal is to reduce model parameters, simplify architectures, and accelerate the inference process. We employed network slimming, a technique that enables us to achieve a streamlined network without introducing excessive complexity. Following this, fine-tuning is performed to restore the original performance, ensuring that the generator's feature extraction and image generation capabilities are preserved. In this context, the original generator model in LW-DCGAN is referred to as the teacher generator, whereas the slimmed-down version is known as the student generator.

In addition, we compute the scaling factor γ_i for the BN layer. This factor directly influences the output of the convolutional layer, and if its value is too small, the overall output of that channel will be biased toward lower values. This implies that, during the forward propagation process, the channel carries less information. The scaling factor γ_i can thus serve as a basis for determining the importance of convolutional channels, as shown in Eq. (8)

$$\gamma_i = s \times \frac{1}{\sqrt{{\sigma_i}^2 + \epsilon}}, \tag{8}$$

where s is a constant scaling parameter applied during normalization, which directly affects the output of the convolution layer. σ^2 is the variance of the i'th channel. ϵ is a very small positive number. Use 10^{-5} as the value of ϵ. γ_i is initialized to 1 and then gradually adjusted through the training process.

In the optimization objective of the generator, the $L1$ regularization term is introduced as a penalty to limit the numerical size of γ_i. An $L1$ regularization term is added to the original loss function to penalize the sum of the absolute values of the scaling factors. The optimization goal is given by the following expression, as shown in Eq. (9)

$$L_{\mathrm{new}} = L_G + \alpha \sum_i |\gamma_i|, \tag{9}$$

where L_G represents the original generator loss function, L_{new} stands for the total loss after introducing the $L1$ regularization term, and α is the coefficient of the $L1$ regularization term, controlling the impact of the regularization term on the total loss. i denotes different channel layers.

Correction of the scaling factor γ_i is given by the following, as shown in Eq. (10)

$$\gamma_i = \mathrm{sign}(\gamma_i) \times \max(|\gamma_i| - \lambda \times l_r, 0), \tag{10}$$

where the sign function converts elements greater than 0 to 1 and those less than 0 to -1, and l_r stands for the learning rate. λ is the $L1$ regularization strength that controls the intensity of scaling factor decay during the correction process.

Subsequently, all γ_i are sorted, and the clipping threshold is selected according to the clipping ratio. Fifty percent of the channels are clipped, and the threshold reflects the middle γ_i value. Then, the scaling factor on the convolution channel of each layer is compared with the threshold. If it is greater than the threshold, the weight parameters of the layer are retained; otherwise, the parameters are set to zero. This step allows many convolution channels with weights of 0 to exist in the model, achieving the sparseness of the model. Record the number of channels and channel numbers that retain weights in each convolutional layer, and redesign the depth of each layer of the network based on the number of channels to obtain a new, more streamlined student generator

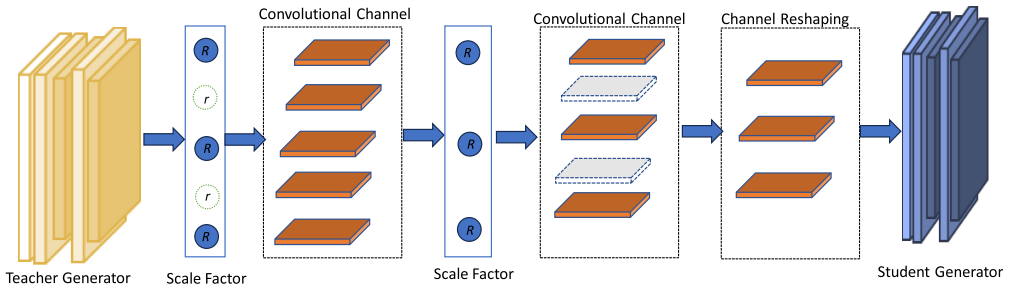


Fig. 7 Slimming process of the generator.

model. The remaining weights in the old model are put into the new model to realize the construction of the student generator.

In the streamlined network, the number of channels in the fully connected layers is reduced from 256 to 128. Consequently, the channel count in the transposed convolution layers is also decreased, leading to a reduction in the number of parameters in each layer and a decrease in the overall network complexity. Figure 7 illustrates the process of network slimming. In Fig. 7, *R* represents the scaling factor greater than the threshold, whereas *r* denotes the scaling factor smaller than the threshold.

Although the algorithm has streamlined the number of channels and network layers, it retains the core functionality necessary to ensure that the network continues to effectively generate high-quality images. The structural details of the slimmed-down LW-DCGAN generator network are outlined in Table 1, which presents the output shapes, operations, convolution kernel sizes, strides, and activation functions utilized at each level.

Table 1 The slimmed-down generator network architecture.

Layer	Output shape	Conv kernel/stride
Input	z_dim, label_dim	—
LabelEncoding	(7, 7, 128)	
Dense	(7, 7, 128)	
Reshape	(7, 7, 128)	—
Conv2DTranspose	(14, 14, 64)	3 × 3/2
BatchNormalization	(14, 14, 64)	—
FPN	(14, 14, 64)	—
ARCM	(14, 14, 64)	—
Conv2DTranspose	(28, 28, 32)	3 × 3/2
BatchNormalization	(28, 28, 32)	—
Conv2DTranspose	(56, 56, 16)	3 × 3/2
BatchNormalization	(56, 56, 16)	—
Conv2DTranspose	(112, 112, 8)	3 × 3/2
BatchNormalization	(112, 112, 8)	—
Segmentation head	(224, 224, 3)	—
Conv2DTranspose	(224, 224, 3)	3 × 3/2
Tanh activation	(224, 224, 3)	—

Table 2 Discriminator network architecture.

Layer	Output shape	Conv kernel/stride
Input	(112, 112, 16)	—
Conv2D	(112, 112, 16)	3 × 3/2
LeakyReLU	(112, 112, 16)	—
Channel attention	(112, 112, 16)	—
Multiply	(112, 112, 16)	—
Conv2D	(56, 56, 32)	3 × 3/2
BatchNormalization	(56, 56, 32)	—
LeakyReLU	(56, 56, 32)	—
Conv2D	(28, 28, 64)	3 × 3/2
BatchNormalization	(28, 28, 64)	—
LeakyReLU	(28, 28, 64)	—
Conv2D	(14, 14, 128)	3 × 3/2
BatchNormalization	(14, 14, 128)	—
LeakyReLU	(14, 14, 128)	—
Flatten	(14, 14, 128)	—

3.3 Discriminator Network

The discriminator plays a crucial role in LW-DCGAN by distinguishing and categorizing images generated by the generator from real images, prompting continuous optimization by the generator. To ensure accurate feature capture from the input data, an attention mechanism is introduced into the discriminator, enhancing the original DCGAN design rather than employing lightweight processing. This attention mechanism improves the accuracy and effectiveness of the discriminator by enabling it to focus on important image regions. The network architecture of the LW-DCGAN discriminator is outlined in Table 2.

3.3.1 *CA module*

For each position x_i, we calculate the attention weights between it and all positions. This attention weight is obtained through linear transformation and dot product operation of input features, as shown in Eq. (11)

$$\text{Attention}(x_i) = \frac{1}{C1}\sum_{j=1}^{c}(W_{\text{query}}(x_i) \cdot W_{\text{key}}(x_j)) \cdot W_{\text{value}}(x_j), \tag{11}$$

where $C1$ is the normalization factor, and c represents the number of channels in the feature map. W_{query}, W_{key}, and W_{value} are weights learned through convolutional operations, used to map the input elements x_i and x_j. Once the attention weights are obtained, we add them to the original features x_i to obtain the weighted feature as shown in Eq. (12)

$$\text{Output}(x_i) = \text{Attention}(x_i) + x_i. \tag{12}$$

The features obtained in the discriminator are connected to a face recognition module. As the preceding convolutional layers have already extracted deep features of facial images, we added a GAP layer. This step performs dimensionality reduction on the feature map, reducing its spatial dimensions to 1 while preserving feature information for each channel. Subsequently, a global average pooling is applied to the weighted features, reducing the spatial dimensions to 1. The result of this global average pooling is given by the following, as shown in Eq. (13)

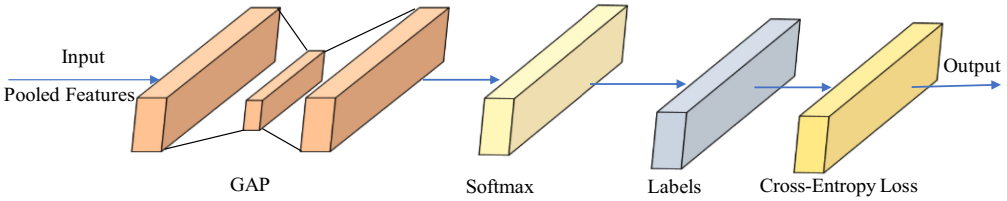


Fig. 8 Architecture of the face recognition module.

$$\text{Pooled Feature}(c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \text{Weighted Feature}(i, j, c), \tag{13}$$

where Weighted Feature(i, j, c) denotes the value of the weighted feature map at position (i, j) and channel c, H and W are the height and width of the weighted feature, respectively. Pooled Feature(c) illustrates the value of the c channel in the resulting feature map after global average pooling, i.e., the weighted average across all positions for that channel.

3.3.2 *Face recognition module*

To simplify the network architecture of the face recognition module and reduce the number of parameters, we utilize pooled features as the input for the face recognition module, directing them to the softmax layer without incorporating an additional fully connected layer. This streamlined design leverages deep features extracted before the discriminator, enabling the face detection module to perform its tasks without unnecessary complexity.

The softmax layer, a standard output layer in deep learning neural networks for multi-class classification problems, transforms the network output into a probability distribution. Predicted probabilities for each category range between 0 and 1, with the sum across all categories equaling 1. During training, the output of the neural network passes through the softmax layer, yielding a predicted probability distribution, which is then compared with the true labels' one-hot encoding. The cross-entropy loss function quantifies the dissimilarity between these two distributions, converting the difference between the network's predictions and the actual labels into a scalar value. This scalar value serves as a metric for evaluating the accuracy of the model's predictions.

Through optimization algorithms such as gradient descent, the optimizer seeks to minimize the cross-entropy loss function, aligning the predictions more closely with the actual labels. This design not only preserves the integrity of feature extraction prior to the discriminator but also makes the entire network architecture more lightweight, making it suitable for deployment in resource-constrained environments. The architecture of the face recognition module is illustrated in Fig. 8.

3.4 Related Loss Functions

3.4.1 *Generator loss function*

LW-DCGAN is designed for the effective generation and recognition of occluded faces. To achieve this, we employ multiple loss functions to guide the training of the generator. Initially, the mean squared error (MSE) loss is used to measure the pixel-level difference between the generated and target images, as shown in Eq. (14)

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (|G(x_i) - y_i|)^2, \tag{14}$$

where $G(x_i)$ signifies the image generated by the generator from input noise vector x_i, y_i represents the corresponding target image for x_i, and N is the number of training samples.

Second, we employ feature matching loss to assess the quality of generated images. Feature matching loss is achieved by comparing the feature representations of generated and real images at the intermediate layer of the discriminator. Feature matching loss is shown in Eq. (15)

$$L_{FM} = \frac{1}{N}\sum_{i=1}^{N} \|D_m(x_i) - D_m(G(z_i))\|^2, \tag{15}$$

where z_i illustrates the input noise vector, $D_m(x_i)$ denotes the output features of discriminator D_m given input x_i, and $G(z_i)$ stands for the generated output by generator G given noise z_i.

Simultaneously, we introduce adversarial loss and use generative adversarial networks to improve the performance of the generator. The adversarial loss drives the generator to learn to generate more realistic images by comparing the output of the generator with the assessments made by the discriminator. The adversarial loss is shown in Eq. (16)

$$L_{\text{AD}} = -\frac{1}{N}\sum_{i=1}^{N} \log(D(G(z_i))). \tag{16}$$

To leverage segmentation label information, we incorporate a cross-entropy loss function into the generator's loss for the segmentation task. Minimizing this loss effectively optimizes the generator's ability to extract segmentation features and predict accurate segmentation results, thereby enhancing image generation using segmentation label information. The cross-entropy loss is presented in Eq. (17)

$$L_{\text{CE}} = -\sum_{K=1}^{K} Y_K \log(P_K), \tag{17}$$

where Y_K represents the true distribution of segmentation labels, indicating the actual segmentation results, whereas P_K is the predicted distribution of the generator, representing the predicted probabilities for each category. K denotes the number of classes.

When the four loss functions are combined, the comprehensive loss function of the LW-DCGAN generator can be obtained. The comprehensive loss is shown in Eq. (18):

$$L_{\text{Generator}} = \lambda 1 \times L_{\text{MSE}} + \lambda 2 \times L_{\text{FM}} + \lambda 3 \times L_{\text{AD}} + \lambda 4 \times L_{\text{CE}}, \tag{18}$$

where $\lambda 1$, $\lambda 2$, $\lambda 3$, and $\lambda 4$ are weight parameters used to balance the contributions of various loss terms.

3.4.2 *Discriminator loss function*

The discriminator in LW-DCGAN employs both binary cross-entropy loss and gradient penalty loss. Binary cross-entropy loss serves to quantify the distinction between generated and real images. The objective of the discriminator is to accurately classify a generated image as false (0) or a real image as true (1). The binary cross-entropy loss L_{BCE} is denoted as in Eq. (19)

$$L_{\text{BCE}} = -\frac{1}{\text{N}}\sum_{i=1}^{\text{N}} [y_i \times \log(D(x_i)) + (1 - y_i) \times \log(1 - D(x_i))]. \tag{19}$$

The gradient penalty loss is applied to enforce the constraint that the gradient norm of the discriminator approaches 1, thereby improving the training stability of the discriminator and the quality of the generated images. The gradient penalty loss is illustrated in Eq. (20)

$$L_{\text{GP}} = \lambda \times [(\|\nabla x D(x)\|_2 - 1)^2], \tag{20}$$

where $\nabla x D(x)$ stands for the gradient of the discriminator D concerning the input x. The overall loss function is a weighted sum of the L_{BCE} and the L_{GP}.

This combination is critical for ensuring that the discriminator network effectively distinguishes between real and fake images while maintaining stable gradients. The comprehensive loss function $L_{\text{Discriminator}}$ is expressed in Eq. (21)

$$L_{\text{Discriminator}} = L_{\text{BCE}} + \alpha \times L_{\text{GP}}, \tag{21}$$

where α is a hyperparameter that balances the contribution of the gradient penalty relative to the L_{BCE}. By minimizing $L_{\text{Discriminator}}$, the discriminator is trained to develop robust recognition capabilities for occluded faces.

3.4.3 *Recognition module loss function*

In the recognition module, we employ cross-entropy loss as the primary loss function to evaluate the classification performance of the model. This widely used loss function effectively gauges the disparity between the probability distribution generated by the model's output and the actual labels. Our goal in this context is for the model to accurately categorize input face images into distinct classes, and cross-entropy loss plays a crucial role in assessing the classification accuracy of the model. At the output layer of the model, we utilize the softmax activation function to transform the initial model output into a class probability distribution.

This activation function ensures that the sum of probabilities for all categories equals 1. Cross-entropy loss then computes the loss between this probability distribution and the distribution of the true labels. Specifically, for a given sample, assuming the raw output of the model is represented as $z = (z_1, z_2, \ldots, z_C)$, where z_i denotes the score for the i'th category, and the actual labels are indicated as $y = (y_1, y_2, \ldots, y_C)$, where y_i conveys the true label for the i'th category, the cross-entropy loss is shown in Eq. (22)

$$L_{\mathrm{CE}} = -\sum_{i=1}^{C} y_i \cdot \log(\mathrm{softmax}(z_i)), \tag{22}$$

where c signifies the number of categories. y_i is the actual label of the category. $\mathrm{softmax}(z_i)$ expresses the predicted probability for the i'th category after passing through the softmax function.

3.5 Training Algorithm of LW-DCGAN

During the training process of LW-DCGAN, the generator G is used to convert the real data images into the generated image G_images. The task of the discriminator D is to distinguish *images* and G_images. z is a random variable, and $pz(z)$ specifies the probability distribution of this random variable z. D_r expresses the prediction result of the discriminator on the real image, and D_f is the prediction result of discriminator for the fake image. m represents the number of samples in a mini-batch during each training iteration. θ_g represents the parameter set of the generator G, whereas θ_d refers to the parameters in the discriminator D. $\nabla\theta_g$ is the gradient of the loss function with respect to θ_g, whereas $\nabla\theta_d$ represents the gradient of the discriminator with respect to θ_d. W_g is the current values of the weights in generator G, whereas W_d expresses the current weight parameters of the discriminator D. η_g defines the learning rate of the generator G, whereas η_d is the learning rate for the discriminator D. Using $\sum_{i=1}^{m} \nabla\theta_g L_{\mathrm{Generator}}(x_i)$, we compute the gradient of the generator loss function for each example x_i, and then sum these gradients to get the total gradient of the generator loss function over the mini-batch. Similarly, $\sum_{i=1}^{m} \nabla\theta d L_{\mathrm{Discriminator}}(x_i)$ calculates the gradient of the discriminator loss function for each sample x_i and then sums these gradients to obtain the total gradient of the discriminator loss function on the mini-batch. This total gradient is used to update the weights of the discriminator to minimize the discriminator loss function. The training algorithm is shown in Table 3.

4 Experimental Analysis

The assessment of LW-DCGAN involved a multi-faceted approach. First, we conducted an ablation experiment on LW-DCGAN to systematically investigate the impact of each algorithmic component on the model's overall performance. Next, we designed generalization experiments to evaluate the model across both training and test datasets. These experiments aimed to assess the model's ability to generalize to new, unseen data, ensuring robust performance in diverse scenarios beyond the training data. Finally, we replaced the generative adversarial network modules in LW-DCGAN with those of GAN, DCGAN, Wasserstein GAN (WGAN-GP), and image-to-image translation with a conditional adversarial network (Pix2Pix). Comparative experiments were then conducted to evaluate the performance of these models against LW-DCGAN.

Table 3 Pseudo-code of LW-DCGAN training algorithm.

Initialize G with random weights

Initialize D with random weights

Input images from dataset F_D

Noise distribution $pz(z)$

While not converged do // Sample mini-batch

$z \sim pz(z)$ // Forward propagation

$G\_\text{images} = G(z; w_g)$

$D_r = D(\text{images}; w_d)$

$D_f = D(G\_\text{images}; w_d)$

// Calculate losses

$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(|G(x_i) - y_i|)^2$

$L_{\text{FM}} = \frac{1}{N}\sum_{i=1}^{N} \|D_m(x_i) - D_m(G(z_i))\|^2$

$L_{\text{AD}} = -\frac{1}{N}\sum_{i=1}^{N} \log(D(G(z_i)))$

$L_{\text{CE}} = -\sum_{K=1}^{\text{K}} Y_K \log(P_K)$

$L_{\text{Generator}} = \lambda_1 \times L_{\text{MSE}} + \lambda 2 \times L_{\text{FM}} + \lambda 3 \times L_{\text{AD}} + \lambda 4 \times L_{\text{CE}}$

$L_{\text{BCE}} = -\frac{1}{\text{N}}\sum_{i=1}^{\text{N}}[y_i \times \log(D(x_i)) + (1 - y_i) \times \log(1 - D(x_i))]$

$L_{\text{GP}} = \lambda \times [(\|\nabla x D(x)\|_2 - 1)^2]$

$L_{\text{Discriminator}} = L_{\text{BCE}} + \alpha \times L_{\text{GP}}$

// Update the generator's weights using the average gradient of the generator's loss

$W_g = W_g - \eta_{\text{g}}\left(\frac{1}{m}\sum_{i=1}^{m} \nabla\theta g L_{\text{Generator}}(x_i)\right)$

// Update the discriminator's weights using the average gradient of the discriminator's loss

$W_d = W_d - \eta_{\text{d}}\left(\frac{1}{m}\sum_{i=1}^{m} \nabla\theta d L_{\text{Discriminator}}(x_i)\right)$

End while

Return G, D, D_f

// Algorithm: Face recognition module with feature extraction

$G_f = \text{GlobalAveragePooling}(D_f)$// Global average pooling

$S_{\text{output}} = \text{Softmax}(G_f)$ // Softmax layer for classification

$L_{\text{CE}} = -\sum_{i=1}^{C} y_i \cdot \log(\text{softmax}(z_i))$ // Cross-entropy loss calculation

// Optimization using backpropagation

Backpropagate(L_{CE})

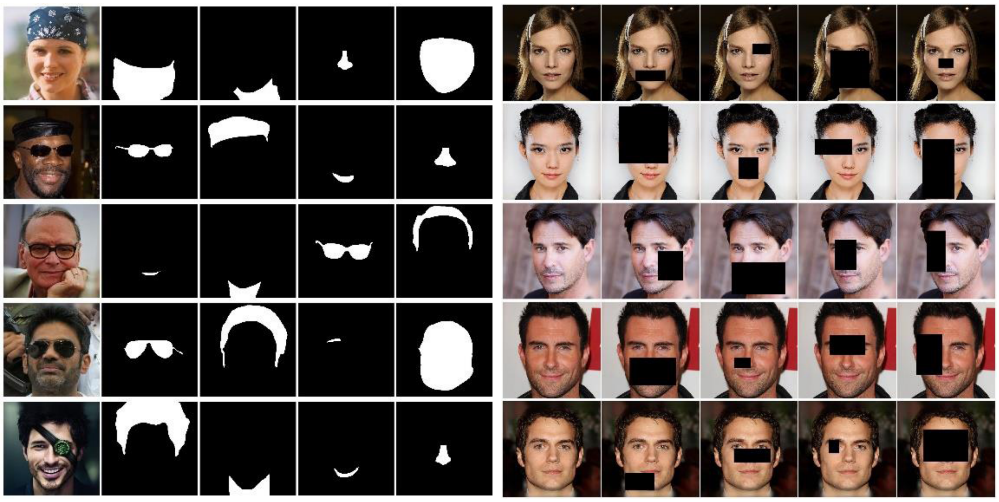
// Update parameters through backpropagation

Return Trained face recognition module

Output: Recognized identity

4.1 Preprocessing of Dataset

The experimental dataset, CelebA-Mask, comprises over 24,000 face images from more than 4000 individuals, each meticulously annotated with detailed facial features such as hair, eyes, mouth, nose, and facial contours. CelebA-Mask, chosen for its multi-label segmentation properties, serves as an ideal foundation for this study.[46]



(a) (b)

Fig. 9 Partial display of the dataset. (a) Set A. (b) Set B.

For training set A, 4000 original face images were selected from this dataset. To address the diverse types of occlusions found in real-world scenarios, 4000 unoccluded images were also included for training set B. During data preparation, variability was introduced by applying element-wise multiplication, adding randomly positioned and sized black occlusions to the original images in training set B. Throughout the training process, batches of data from both datasets A and B were randomly selected. Segmentation labels from dataset A were fed into the generator to enrich facial structural information, whereas pixel-level details from dataset B were directly used as input for the generator. A fivefold cross-validation approach was employed, dividing the entire dataset into five subsets. Four subsets were used for training, whereas the remaining subset served as the test set. This process was repeated five times, with each subset serving as the test set once, ensuring a comprehensive evaluation. The test set involved in each training iteration is referred to as testing set 1. Figure 9 displays samples from the dataset.

To comprehensively evaluate the performance of LW-DCGAN in the occluded face recognition tasks, we additionally selected three widely used datasets as test sets. These three datasets are no longer involved in the training process and are only used as result tests. The Caltech occluded faces in the wild (COFW) dataset is specifically designed to study occluded faces, featuring numerous images with various occlusions such as hats, glasses, and hands, which facilitates testing the performance of the model under complex occlusion conditions.[47] The LFW dataset includes images with occlusions such as glasses and hats and is primarily used to assess the model's performance under natural occlusion conditions.[48] The masked face recognition v2 (MFR2) dataset, introduced during the COVID-19 pandemic, addresses face recognition challenges with masks. It includes images with various mask types, such as medical masks, N95 masks, and cloth masks, as well as other occluders such as sunglasses and hats, simulating partially occluded facial scenes in real-world scenarios.[49] Table 4 displays the varying occlusion ratios across the datasets.

Table 4 Data set distribution.

Occlusion rate	0%	10% to 15%	15% to 30%	30% to 50%	50% to 80%
Training set	1000	1800	1200	1400	1000
Testing set 1	300	300	400	350	250
COFW	50	120	200	100	30
LFW	280	100	70	50	0
MFR2	160	80	100	100	60

Table 5 Experimental parameters.

Parameter	Value	Parameter	Value
Weight_decay	0.0001	Gradient penalty parameter	10
Batch size	64	Mask pool size	5
Epoch	200	Generator learning rate	0.001
Height_stride	16	Discriminator learning rate	0.0005
Width_stride	16	Momentum	0.9
Noise vector dimension	100	Detection_max_instances	1024
Discriminator optimizer	Adam(lr=0.0002, beta_1=0.5)	Detection_min_confidence	0.128
Generator optimizer	Adam(lr=0.0002, beta_1=0.5)	Detection_NMS_Threshold	0.3
Image size	64×64	Number of feature channels	64

4.2 Experimental Setup and Parameters

The LW-DCGAN algorithm model developed in this study was trained using the TensorFlow 2.5.0 framework, with acceleration provided by an NVIDIA GeForce RTX 2080 Ti GPU. The training was conducted on a computer system equipped with an Intel(R) Core(TM) i5-11320H @ 3.20 GHz processor, Intel(R) UHD Graphics 630 adapter, 16 GB of memory, and a 64-bit operating system. The model was programmed in Python 3.8.3 using the PyCharm 2021 integrated development environment.

The experimental environment offered ample computing resources and stable software support, ensuring the efficient training of the LW-DCGAN model and the achievement of accurate results. The parameter settings used in the experiment are detailed in Table 5.

To better utilize the face segmentation label information during the training process of training set A, an additional channel is added at the input layer to incorporate the segmentation labels corresponding to the facial images. A branch is then introduced to handle this segmentation label input, which includes a 1×1 convolution layer for mapping label features that are subsequently fused with the main network.

Considering the added task of predicting segmentation, a 1×1 convolution prediction branch is inserted just before the output of the final transposed convolution layer. This branch has the same number of output channels as the segmentation label channel and is designed to extract segmentation feature maps. During the initialization phase, the additional channel parameters of the generator are set to a non-trainable state, effectively freezing these parameters. Throughout the training loop, by selectively enabling or restoring the training status of these additional channel parameters, we can flexibly control the extent to which the generator utilizes multi-label information when processing different dataset groups. For datasets that do not use multi-label segmentation, the focus remains on mean squared error loss, feature matching loss, and adversarial loss.

4.3 Ablation Experiment

Ablation experiments help determine the relative importance of various components within the model. By comparing performance differences after removing specific components, we can identify which factors contribute the most to overall model performance. The evaluation metrics used in this experiment include accuracy, recall, SSIM, and PSNR. Accuracy, a commonly used metric in classification models, represents the proportion of correctly classified samples to the total number of samples. It is calculated as shown in Eq. (23)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{23}$$

where TP denotes the number of samples correctly classified as positive, indicating the true positive count. FP represents the count of samples incorrectly classified as positive, despite being

negatives in reality. TN refers to the number of samples correctly classified as negatives, i.e., true negatives. FN signifies the count of samples incorrectly classified as negatives, despite being positives in reality.

Recall rate refers to the proportion of samples that the model correctly predicts as positive samples among all the samples that are positive. It is as shown in Eq. (24)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{24}$$

The SSIM is employed to gauge the similarity between two images, with a value closer to 1 indicating higher similarity. It is represented by Eq. (25)

$$\text{SSIM}(I_1, I_2) = \frac{(2\mu_1\mu_2 + C_1) \times (2\sigma_{12} + C_2)}{({\mu_1}^2 + {\mu_2}^2 + C_1) \times ({\sigma_1}^2 + {\sigma_2}^2 + C_2)}, \tag{25}$$

where I_1 and I_2 represent two images, respectively. μ_1 and μ_2 illustrate the respective mean of the two images. σ_1 and σ_2 express the standard deviation of the two images. σ_{12} symbolizes the covariance of the two images. C_1 and C_2 are constants introduced for stability. PSNR is a metric used to measure image quality, and a higher value indicates better image quality. It is given by the following, as shown in Eq. (26)

$$\text{PSNR}(I_1, I_2) = 10 \log_{10}\left(\frac{255^2}{\text{MSE}(I_1, I_2)}\right), \tag{26}$$

where $\text{MSE}(I_1, I_2)$ depicts the mean square error, that is, the degree of difference between the two images. SSIM and PSNR measure the image generation capabilities of the generative adversarial network module in LW-DCGAN.

The experiment is divided into three groups. In experiment group A, the full architecture of LW-DCGAN is maintained, including both the FPN and ARCM components. In experiment group B, the FPN is removed from LW-DCGAN while retaining ARCM, resulting in a network denoted as LW-DCGAN (ARCM). In experiment group C, ARCM is excluded whereas FPN is retained, creating the network referred to as LW-DCGAN (FPN). All three groups utilize the same training dataset and apply a cosine annealing learning rate decay strategy. Figure 10 illustrates the changes in each metric from 0 to 200 epochs across these ablation experiments.

By comparing the performance indicators of LW-DCGAN, LW-DCGAN (ARCM), and LW-DCGAN (FPN), several valuable conclusions can be drawn regarding the roles of FPN and ARCM in LW-DCGAN. First, LW-DCGAN demonstrated the highest performance, with an accuracy rate of 88%, a recall rate of 90%, an SSIM index of 0.877, and a PSNR of 28.3 after 200 epochs. The metrics for LW-DCGAN (ARCM) declined compared with LW-DCGAN, showing a 6% decrease in accuracy, a 4% decrease in recall, a drop of 0.035 in SSIM, and a reduction of 0.6 in PSNR. This suggests that FPN has a significantly positive impact on the performance of image generation and classification tasks. In contrast, when ARCM was removed while retaining FPN in LW-DCGAN (FPN), accuracy decreased by 13%, recall by 10%, SSIM by 0.082, and PSNR by 0.95. These significant declines in metrics indicate that the attention mechanism plays a crucial role in the model's performance. Overall, LW-DCGAN, incorporating both FPN and ARCM, achieved superior performance.

4.4 Generalization Experiment

In the generalization experiment, we meticulously recorded the accuracy, recall, generator loss, discriminator loss, and changes in PSNR and SSIM for both the training set and testing set 1 throughout the iteration process. In addition, the final performance metrics of the model, including accuracy, recall, SSIM, and PSNR, were evaluated on widely used datasets such as COFW, LFW, and MFR2. This analysis primarily serves to assess how well the model generalizes to unseen data. Figure 11 illustrates the accuracy and recall of the model on the training set and testing set 1 during the iterative training process.

As the number of epochs increased, both the training set accuracy and testing set 1 accuracy showed gradual improvement. Initially, at epoch 0, the accuracy was relatively low, but as training progressed, it steadily increased until reaching a saturation point. The accuracy on the training set rose from 0.20 to 0.88, whereas the accuracy on testing set 1, after some initial

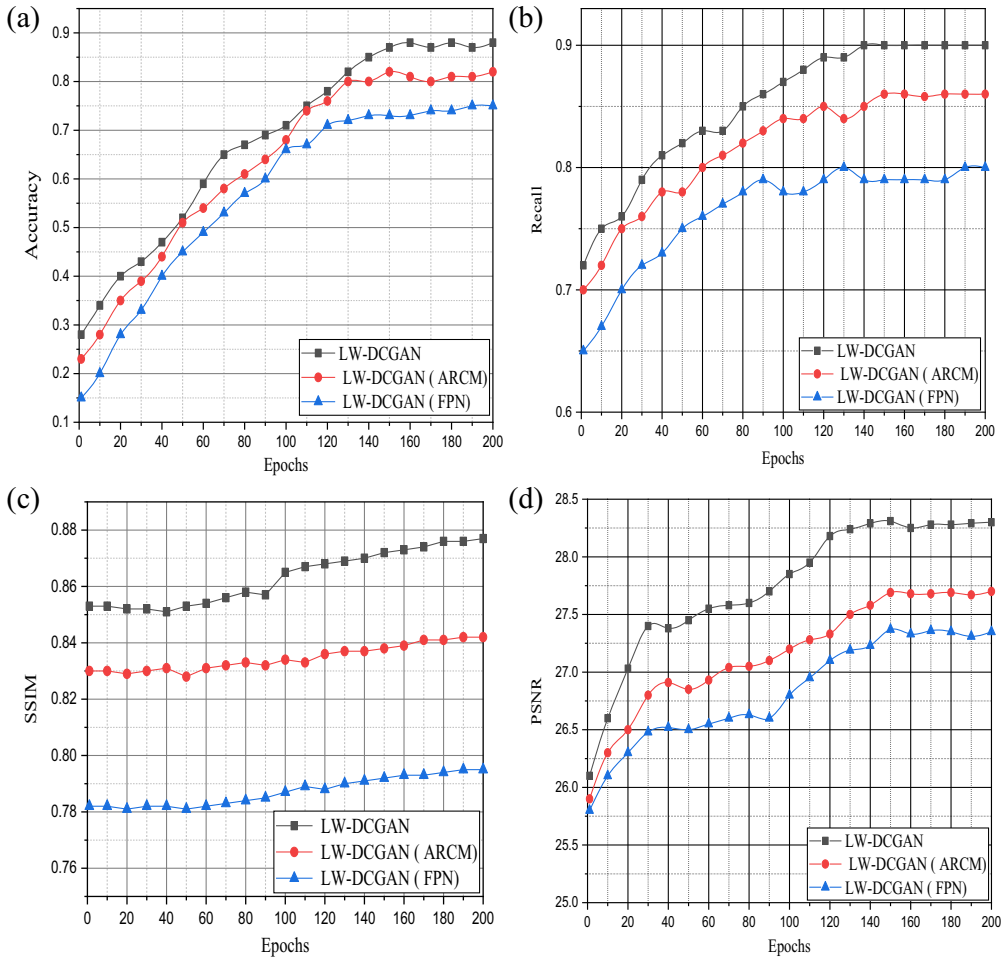


Fig. 10 Comparison of evaluation metrics in ablation experiments. (a) Accuracy. (b) Recall. (c) SSIM. (d) PSNR.

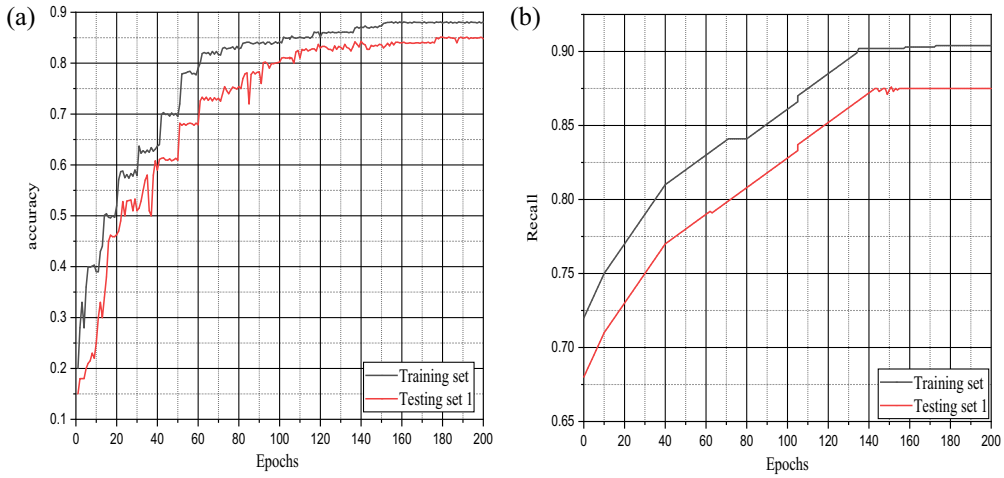


Fig. 11 Accuracy and recall in generalization experiment. (a) Accuracy. (b) Recall.

fluctuations, showed a significant upward trend, increasing from 0.15 to 0.85. Similarly, recall also improved for both datasets, with the training set stabilizing at 0.9 and testing set 1 at 0.87. This suggests that the LW-DCGAN model demonstrates a strong generalization ability in recognizing occluded faces, achieving high accuracy and recall on testing set 1. In addition, a comparison of accuracy and recall between the training set and testing set 1 indicates that the model

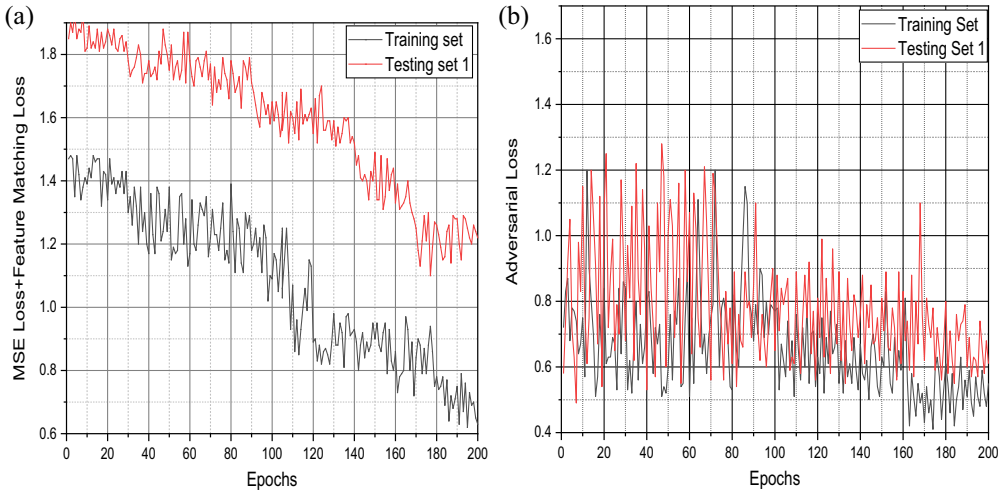


Fig. 12 Generator loss. (a) MSE loss and feature matching loss. (b) Adversarial loss.

did not exhibit significant signs of overfitting. The experiment further evaluates the LW-DCGAN model using both generator loss and discriminator loss. Figure 12 illustrates the generator loss.

The data show a gradual decrease in both mean square error loss and feature matching loss as training progresses. On the training set, the generator's loss decreases from an initial value of 1.47 to 0.93. This trend aligns with the characteristics of LW-DCGAN, indicating that the model gradually learns more effective generation and discrimination strategies during training. Meanwhile, adversarial loss fluctuates throughout the entire training process, reflecting the oscillations caused by the ongoing competition between the generator and discriminator. However, as shown in Fig. 12, adversarial loss exhibits an overall downward trend.

To form the comprehensive loss function for the discriminator, we combine binary cross-entropy loss with gradient penalty loss. The resulting discriminator loss rates are recorded in Fig. 13, which illustrates an overall fluctuating downward trend in the loss rates for both the training set and testing set 1.

The variation curves of PSNR and SSIM on the training set and testing set 1 are illustrated in Fig. 14. From the graph, it is evident that although PSNR experiences significant fluctuations, it shows an overall upward trend. Meanwhile, SSIM gradually increases in both the training set and

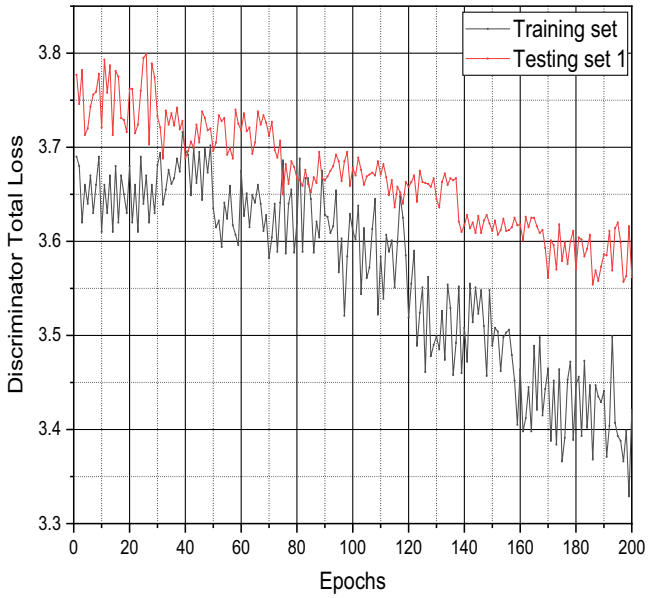


Fig. 13 Discriminator loss.

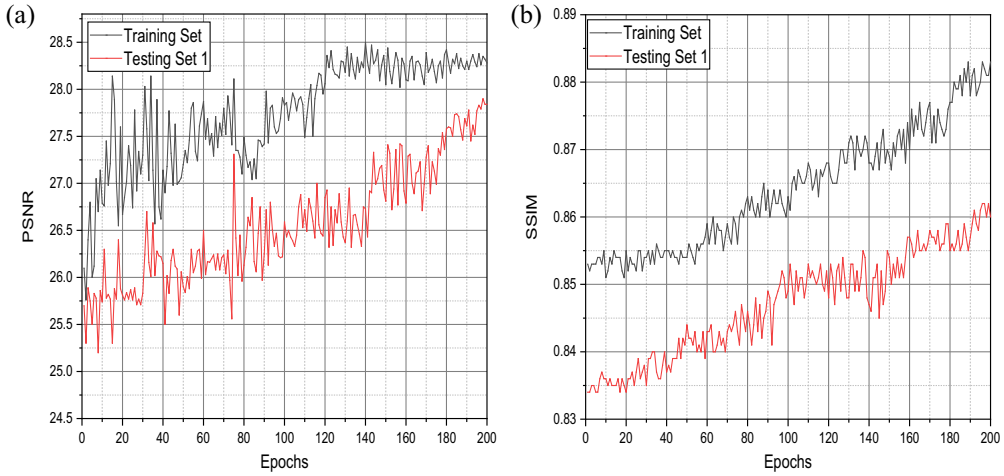


Fig. 14 PSNR and SSIM variation curve. (a) PSNR. (b) SSIM.

Table 6 Performance of LW-DCGAN on different datasets.

Data set	Accuracy (%)	Recall (%)	PSNR	SSIM
Training set	88	90	28.3	0.881
Testing set 1	85	87	27.8	0.863
COFW	82	87	27.6	0.867
LFW	94	91	28.5	0.874
MFR2	86	88	27.9	0.877

testing set 1, indicating the progressive enhancement of image quality throughout the training process.

To further verify the generalization ability of LW-DCGAN, we evaluated the performance of the model on COFW, LFW, and MFR2. Table 6 shows the comparison results of LW-DCGAN on the training set, testing set 1, COFW, LFW, and MFR2 in terms of accuracy, recall, SSIM, and PSNR.

On the COFW dataset, which includes complex occlusions, the model achieves an accuracy of 82% and a recall of 87%, with a PSNR of 27.6 and an SSIM of 0.867, demonstrating robustness in challenging conditions. On the LFW dataset, featuring natural occlusions, the model records a high accuracy of 94% and a recall of 91%, with a PSNR of 28.5 and an SSIM of 0.874. Finally, on the MFR2 dataset, which focuses on occlusions from masks, the model performs with an accuracy of 86%, a recall of 88%, a PSNR of 27.9, and an SSIM of 0.877. Overall, these results validate LW-DCGAN's strong performance and adaptability across diverse occluded facial recognition tasks.

4.5 Comparative Experiment

The performance of the LW-DCGAN algorithm was further evaluated through comparison experiments with other GAN variants, including GAN, DCGAN, WGAN-GP, and the Pix2Pix network. Comparing LW-DCGAN with GAN serves as a benchmark to assess its lightweight face recognition capabilities, given GAN's foundational role in generative models. The comparison with DCGAN allows for an analysis of whether LW-DCGAN outperforms more complex deep convolutional architectures. WGAN-GP, which addresses issues such as training instability and mode collapse through improved loss functions, is compared with LW-DCGAN to evaluate the impact of a lightweight design on training stability. Finally, Pix2Pix, a conditional generative adversarial network for image translation, is compared with LW-DCGAN to

Table 7 Comparison of adversarial network parameter designs.

Parameter/model	GAN	DCGAN	LW-DCGAN	WGAN-GP	Pix2Pix
Noise distribution	Uniform distribution [−1, 1]	Uniform distribution [−1, 1]	Uniform distribution [−1, 1]	Gaussian distribution [0, 1]	Uniform distribution [−1, 1]
Activation function	ReLU	LeakyReLU	ReLU Leaky ReLU	Leaky ReLU	ReLU
Batch normalization	No	Yes	Yes	Yes	Yes
Generator loss function	Cross-entropy loss	Cross-entropy loss	Composite loss	Wasserstein loss	*L*1 loss
Discriminator loss function	Cross-entropy loss	Cross-entropy loss	Composite loss	Wasserstein loss	Cross-entropy loss
Gradient clipping parameter	N/A	N/A	N/A	0.01	N/A
L2 regularization parameter	0.0001	0.0001	0.0001	0.0001	0.0001
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001
Momentum	0.9	0.9	0.9	0.9	0.9
Batch size	64	64	64	64	64
Convolutional layers	Yes	Yes	Yes	Yes	Yes
Max pooling layers	N/A	Yes	Yes	N/A	Yes
Convolutional kernels	(3 × 3)	(3 × 3)	(3 × 3)	(4 × 4)	(3 × 3)

determine whether the latter excels in occluded face recognition tasks. The parameter settings for each model are provided in Table 7.

To better compare the performance of LW-DCGAN in face recognition, we selected FaceNet, ArcFace, VGGFace, and SphereFace (based on ResNet-64) as benchmarks. FaceNet is recognized for its robustness in embedding space optimization using triplet loss, ArcFace for its superior inter-class separability through angular margins, VGGFace as a well-established benchmark for evaluating general recognition performance, and SphereFace for its enhanced angle-based inter-class separation. This evaluation helps position LW-DCGAN relative to these established models. The experimental parameters of the four models are shown in Table 8.

To intuitively evaluate the performance of various networks in occluded face recognition, we primarily compare the models based on accuracy and recall. To assess the convergence and stability of different models and observe the variation of these metrics during the training process, the same training dataset and learning rate cosine annealing strategy are employed for each model. Figure 15 illustrates the variations in accuracy and recall for different models from epoch 0 to 200.

In comparative experiments, LW-DCGAN demonstrates significant superiority in training accuracy, achieving a final accuracy of 88% and a recall rate of 90%. Among the GAN-related models, DCGAN and WGAN-GP perform relatively well, with accuracies of 85% and recall rates of 87% and 88%, respectively. However, GAN and Pix2Pix show poor performance. Among the face recognition models—FaceNet, ArcFace, VGGFace, and SphereFace (based on ResNet-64)—SphereFace outperforms the others, achieving an accuracy of 80% and a recall rate of 83%. In contrast, VGGFace performs the worst on datasets with severe occlusion. To further illustrate the advantage of LW-DCGAN in a network scale, we compare these

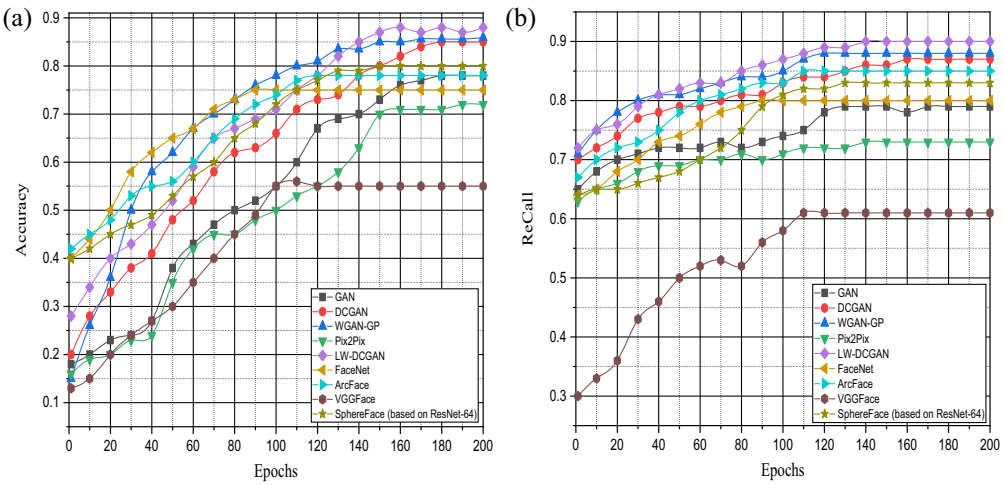


Fig. 15 Comparison of network models. (a) Accuracy. (b) Recall.

Table 8 Comparison of face recognition model parameter designs.

Parameter/model	FaceNet	ArcFace	VGGFace	SphereFace (based on ResNet-64)
Loss function	Triplet loss	Additive angular margin loss	Softmax loss	Angular softmax loss
Embedding size	128	512	4096	512
Activation function	ReLU	ReLU	ReLU	ReLU
Batch normalization	Yes	Yes	Yes	Yes
Optimizer	Adam	SGD	SGD	SGD
Learning rate	0.001	0.01	0.01	0.01
Weight decay	1×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Momentum	0.9	0.9	0.9	0.9
Batch size	64	64	64	64
Data augmentation	Yes	Yes	Yes	Yes
Dropout rate	0.5	0.4	0.5	0.4

models in terms of model memory, parameters, and inference time. All values are shown in Table 9.

From the performance metrics listed in the table, it is evident that after the optimization, the memory usage of the LW-DCGAN generator model is reduced by 4.6 MB compared with the DCGAN, with a decrease of 1M in the number of parameters and a reduction of 0.07 s in inference time.

5 Discussion

In this study, we introduced the lightweight deep convolutional generative adversarial network (LW-DCGAN), specifically designed to address the challenge of recognizing occluded faces. Our research included an extensive evaluation of LW-DCGAN's capabilities through various experimental analyses, underscoring its potential for practical deployment.

LW-DCGAN represents a significant advancement in occluded face recognition technology. It overcomes the limitations of earlier methods by employing a unique network architecture and

Table 9 Comparison of model size and inference time.

Model	Metric	Value
LW-DCGAN	Generator model memory	7.3 MB
	Discriminator model memory	26.3 MB
	Generator model parameters	1.9M
	Discriminator model parameters	7.15M
	Inference time	0.18S
DCGAN	Generator model memory	11.9 MB
	Discriminator model memory	26.3 MB
	Generator model parameters	2.9M
	Discriminator model parameters	7.16M
	Inference time	0.25S
GAN	Generator model memory	7.9 MB
	Discriminator model memory	13.1 MB
	Generator model parameters	1.8M
	Discriminator model parameters	3.1M
	Inference time	0.22S
Pix2Pix	Generator model memory	23.5 MB
	Discriminator model memory	30.1 MB
	Generator model parameters	6.3M
	Discriminator model parameters	7.2M
	Inference time	0.42S
WGAN-GP	Generator model memory	16.1 MB
	Discriminator model memory	25.3 MB
	Generator model parameters	3.4M
	Discriminator model parameters	5.9M
	Inference time	0.35S
FaceNet	Model memory	88.2 MB
	Model parameters	22.8M
	Inference time	0.15S
ArcFace	Model memory	271 MB
	Model parameters	68.1M
	Inference time	0.09S
VGGFace	Model memory	552 MB
	Model parameters	138.3 M
	Inference time	0.2S
SphereFace (based on ResNet-64)	Model memory	96.2 MB
	Model parameters	24.1M
	Inference time	0.08S

innovative algorithms. The multi-layer architecture excels at extracting detailed features across different scales, improving the model's ability to handle diverse and complex datasets. Importantly, the lightweight framework enhances efficiency compared with conventional deep learning models, reducing the dependency on high-end computational resources and enabling deployment on devices with limited processing power, which is crucial for real-time applications.

However, deploying LW-DCGAN also presents challenges that need to be addressed to maximize its utility. The model's performance is closely tied to the dataset used for training and testing. Although the CelebA-Mask dataset includes a wide range of occluded facial images, it does not cover all possible occlusion scenarios encountered in real-world settings. Although LW-DCGAN performs well with common occlusions, its effectiveness diminishes with rarer or more complex types. This limitation suggests the need for further robustness testing and the use of advanced data augmentation techniques to improve the model's applicability across various occlusion conditions and severity levels. In addition, whereas LW-DCGAN is designed to be more efficient than traditional models, the computational resources required during training are still considerable, which could hinder its widespread adoption, particularly in resource-limited environments.

6 Conclusion

This paper presents LW-DCGAN, a specialized generative adversarial network developed for occluded face recognition. LW-DCGAN utilizes a streamlined convolutional network, enhanced with feature pyramids and attention mechanisms, to generate high-quality images while maintaining reduced model complexity. Our ablation studies confirmed that the FPN and ARCM components significantly enhance LW-DCGAN's performance in recognizing occluded faces. Generalization tests further validated its effectiveness across multiple datasets, including COFW, LFW, and MFR2. Comparative experiments against GAN, DCGAN, WGAN-GP, Pix2Pix, and popular face recognition models such as FaceNet, ArcFace, VGGFace, and SphereFace highlighted LW-DCGAN's superior accuracy, recall rates, and smaller model size. In summary, LW-DCGAN offers a robust and scalable solution for occluded face recognition, with potential applications in broader image generation contexts. Future work will focus on optimizing LW-DCGAN for the rapid and precise classification of various occluded or blurred dynamic visual data streams.

Code and Data Availability

Some or all data, models, or codes generated or used during the study are available from the corresponding author upon request.

Acknowledgments

This work was partly supported by the Key R&D projects in Henan Province (Grant No. 241111211800), the Key Scientific and Technological Project of Henan Province (Grant Nos. 232102111128, 222102210098, 222102320181, 212102210431, and 212102310087), in part by the Major Special Project of Xinxiang City (Grant No. 21ZD003), in part by the Key Scientific Research Projects of Colleges and Universities in Henan Province (Grant Nos. 23B520003, 21A520001, and 20A520013), and in part by the Henan Province Postdoctoral Support Program (Grant No. HN2022165).

References

1. I. Goodfellow et al., "Generative adversarial nets," *Adv. Neural Inf. Processing Syst.* **27**, 2672–2680 (2014).
2. A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv:1511.06434 (2015).
3. T. Karras et al., "Progressive growing of GANs for improved quality, stability, and variation," arXiv:1710.10196 (2017).
4. G. Luo et al., "Geometry sampling-based adaption to DCGAN for 3D face generation," *Sensors* **23**(4), 1937 (2023).

5. M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.,* pp. 214–223 (2017).
6. P. Isola, J. Y. Zhu, and T. Zhou, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.,* pp. 1125–1134 (2017).
7. J. Wright et al., "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(2), 210–227 (2008).
8. L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: which helps face recognition?," in *Proc. IEEE Int. Conf. Comput. Vision*, pp. 471–478 (2011).
9. W. Ou et al., "Robust face recognition via occlusion dictionary learning," *Pattern Recognit.* **47**(4), 1559–1572 (2014).
10. H. Zheng et al., "Laplacian-uniform mixture-driven iterative robust coding with applications to face recognition against dense errors," *IEEE Trans. Neural Network Learn. Syst.* **31**(9), 3620–3633 (2019).
11. I. Q. Mundial et al., "Towards facial recognition problem in COVID-19 pandemic," in *Proc. IEEE 4th Int. Conf. Electr., Telecommun. Comput. Eng. (ELTICOM)*, pp. 210–214 (2020).
12. H. N. Vu, M. H. Nguyen, and C. Pham, "Masked face recognition with convolutional neural networks and local binary patterns," *Appl. Intell.* **52**(5), 5497–5512 (2022).
13. D. Montero et al., "Boosting masked face recognition with multi-task ArcFace," arXiv:2104.09874 (2021).
14. W. Hariri, "Efficient masked face recognition method during the COVID-19 pandemic," *Signal Image Video Process.* **16**(3), 605–612 (2022).
15. R. Golwalkar and N. Mehendale, "Masked-face recognition using deep metric learning and FaceMaskNet-21," *Appl. Intell.* **52**(11), 13268–13279 (2022).
16. Y. Zhang, X. Peng, and Y. Guo, "Lightweight network for masked face recognition based on improved dual attention mechanism," in *Proc. IEEE Int. Conf. Mechatron. and Autom. (ICMA)*, pp. 1621–1626 (2023).
17. B. Huang et al., "PLFace: progressive learning for face recognition with mask bias," *Pattern Recognit.* **135**, 109142 (2023).
18. Y. Ge et al., "Masked face recognition with convolutional visual self-attention network," *Neurocomputing* **518**, 496–506 (2023).
19. W. C. Cheng, H. C. Hsiao, and L. H. Li, "Deep learning mask face recognition with annealing mechanism," *Appl. Sci.* **13**(2), 732 (2023).
20. M. Zhong et al., "MaskDUF: data uncertainty learning in masked face recognition with mask uncertainty fluctuation," *Expert Syst. Appl.* **238**, 121995 (2024).
21. M. O. Faruque, M. R. Islam, and M. T. Islam, "Advanced masked face recognition using robust and light weight deep learning model," *Int. J. Comput. Appl.* **975**, 8887 (2024).
22. S. Sharma et al., "KInsight: a robust framework for masked face recognition," *Braz. Arch. Biol. Technol.* **67**, e24230917 (2024).
23. C. Li et al., "Look through masks: towards masked face recognition with de-occlusion distillation," in *Proc. 28th ACM Int. Conf. Multimedia*, pp. 3016–3024 (2020).
24. Y. Fu et al., "LE-GAN: unsupervised low-light image enhancement network using attention module and identity invariant loss," *Knowl.-Based Syst.* **240**, 108010 (2022).
25. B. Chen et al., "Locally GAN-generated face detection based on an improved Xception," *Inf. Sci.* **572**, 16–28 (2021).
26. X. Zhang et al., "DE-GAN: domain embedded GAN for high quality face image inpainting," *Pattern Recognit.* **124**, 108415 (2022).
27. J. Lin, Y. Li, and G. Yang, "FPGAN: face de-identification method with generative adversarial networks for social robots," *Neural Network* **133**, 132–147 (2021).
28. Y. Zhang, M. Ding, and Y. Bai, "Detecting small faces in the wild based on generative adversarial network and contextual information," *Pattern Recognit.* **94**, 74–86 (2019).
29. D. S. Trigueros, L. Meng, and M. Hartnett, "Generating photo-realistic training data to improve face recognition accuracy," *Neural Network* **134**, 86–94 (2021).
30. X. Yang et al., "Semantic face completion based on DCGAN with dual-discriminator," in *Proc. 2021 7th Annu. Int. Conf. Network and Inf. Syst. for Comput. (ICNISC)*, pp. 10–14 (2021).
31. T. P. Hong et al., "Conditional-GAN-based face inpainting approaches with symmetry and view-degree utilization," *IEEE Access* **8**, 193459–193470 (2020).
32. F. Huang et al., "Cyclic style generative adversarial network for near infrared and visible light face recognition," *Appl. Soft Comput.* **150**, 111096 (2024).
33. G. Andrew and Z. Menglong, "MobileNets: efficient convolutional neural networks for mobile vision applications," **10**, 151 (2017).
34. M. Sandler et al., "MobileNetv2: inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 4510–4520 (2018).
35. A. Howard et al., "Searching for mobilenetv3," in *Proc. IEEE/CVF Int. Conf. Comput. Vision*, pp. 1314–1324 (2019).

36. X. Zhang et al., "ShuffleNet: an extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 6848–6856 (2018).
37. N. Ma et al., "ShuffleNet v2: practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, pp. 116–131 (2018).
38. M. Tan and Q. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.,* PMLR, pp. 6105–6114 (2019).
39. K. Han et al., "GhostNet: more features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.*, pp. 1580–1589 (2020).
40. J. Liu et al., "FDDWNet: a lightweight convolutional neural network for real-time semantic segmentation," in *Proc. ICASSP 2020-2020 IEEE Int. Conf. Acoust. Speech and Signal Process. (ICASSP)*, IEEE, pp. 2373–2377 (2020).
41. Y. Chen et al., "Mobile-former: bridging MobileNet and transformer," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.*, IEEE pp. 5270–5279 (2022).
42. Z. Lyu et al., "A GPU-free real-time object detection method for apron surveillance video based on quantized MobileNet-SSD," *IET Image Process.* **16**(8), 2196–2209 (2022).
43. P. S. P. Kavyashree and M. El-Sharkawy, "Compressed MobileNet v3: a light weight variant for resource-constrained platforms," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop and Conf. (CCWC),* pp. 0104–0107 (2021).
44. H. Shi, Q. Zhou, and Y. Ni, "DPNET: dual-path network for efficient object detection with lightweight self-attention," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 771–775 (2022).
45. L. Jia et al., "MobileNet-CA-YOLO: an improved YOLOv7 based on the MobileNetV3 and attention mechanism for rice pests and diseases detection," *Agriculture* **13**(7), 1285 (2023).
46. C. H. Lee et al., "MaskGAN: towards diverse and interactive facial image manipulation," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.*, pp. 5549–5558 (2020).
47. X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *Proc. IEEE Int. Conf. Compute. Vision*, pp. 1513–1520 (2013).
48. G. B. Huang et al., "Labeled faces in the wild: a database for studying face recognition in unconstrained environments," in *Workshop Faces in 'Real-Life' Images: Detect., Alignment, and Recognit.* (2008).
49. F. Boutros et al., "MFR 2021: masked face recognition competition," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, pp. 1–10 (2021).

Yingying Lv received her BS degree from the School of Information Engineering, Henan University of Science and Technology, China, in 2008 and her MS degree from the School of Information Engineering, Zhengzhou University, China, in 2011. She is currently a lecturer at the School of Computer Science and Technology, Henan Institute of Science and Technology. Her research interests include deep learning, neural networks, and intelligent computing.

Jianping Wang received his BS degree from the Department of Computer Science and Technology, Shaanxi Normal University, Xi'an, China, in 2004; his MS degree from the Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2011; and his PhD in information and communication engineering from Wuhan University of Technology, Wuhan, China, in 2019. He completed his postdoctoral research in control science and engineering at Henan University of Science and Technology, Henan, China, in 2024. He is currently a professor and the deputy dean at the School of Computer Science and Technology, Henan Institute of Science and Technology. His research interests lie in the areas of intelligent computing, software-defined networks, and wireless sensor networks.

Guohong Gao is currently a professor at the School of Information Engineering, Henan Institute of Science and Technology. His research interest includes wireless sensor networks.

Qian Li is currently a teacher at the School of Information Engineering, Henan Institute of Science and Technology. Her main research interest includes big data.