Neural Networks 152 (2022) 487–498

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

GIU-GANs: Global Information Utilization for Generative Adversarial Networks

Yongqi Tian [a,c], Xueyuan Gong [b,*], Jialin Tang [c,d], Binghua Su [a,c], Xiaoxiang Liu [b], Xinyuan Zhang [b]

[a] School of Optoelectronics, Beijing Institute of Technology, Beijing, China
[b] School of Intelligent Systems Science and Engineering, Jinan University, Zhuhai, China
[c] School of Information Technology, Beijing Institute of Technology, Zhuhai, China
[d] Faculty of Data Science, City University of Macau, China

ARTICLE INFO

Article history:
Received 13 September 2021
Received in revised form 16 March 2022
Accepted 15 May 2022
Available online 21 May 2022

Keywords:
Image generation
Generative Adversarial Networks
Global Information Utilization
Involution
Representative Batch Normalization

ABSTRACT

Recently, with the rapid development of artificial intelligence, image generation based on deep learning has advanced significantly. Image generation based on Generative Adversarial Networks (GANs) is a promising study. However, because convolutions are limited by spatial-agnostic and channel-specific, features extracted by conventional GANs based on convolution are constrained. Therefore, GANs cannot capture in-depth details per image. Moreover, straightforwardly stacking of convolutions causes too many parameters and layers in GANs, yielding a high overfitting risk. To overcome the abovementioned limitations, in this study, we propose a GANs called GIU-GANs (where Global Information Utilization: GIU). GIU-GANs leverages a new module called the GIU module, which integrates the squeeze-and-excitation module and involution to focus on global information via the channel attention mechanism, enhancing the generated image quality. Moreover, Batch Normalization (BN) inevitably ignores the representation differences among noise sampled by the generator and thus degrades the generated image quality. Thus, we introduce the representative BN to the GANs' architecture. The CIFAR-10 and CelebA datasets are employed to demonstrate the effectiveness of the proposed model. Numerous experiments indicate that the proposed model achieves state-of-the-art performance.

1. Introduction

Image generation is crucial and challenging in the Computer Vision field. With the development of deep learning, there has been remarkable progress in this field. In 2014, the application of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) for image generation was reported with promising results. Then, GANs based on deep Convolutional Neural Networks (CNNs) (Radford, Metz, & Chintala, 2016) improved the generated image quality. Since then, CNNs have been the core of GANs. However, convolution kernels have two remarkable properties, spatial-agnostic and channel-specific (Li et al., 2021), which contribute to their widespread yet pose some issues.

Spatial-agnostic means that a convolution kernel produces the same output no matter its location in an image. As is well known, the size of a convolution kernel is $C_o \times C_i \times K \times K$, where C_i and C_o are the numbers of input and output channels, respectively, and K is the kernel size. Owing to spatial-agnostic, a convolution kernel has the same parameters in different regions of an image, which reduces computation. However, this constrains the receptive field of convolution, which makes it difficult for models to capture the mathematical relationships of spatially distant locations and thus deprives the ability of convolution to extract global information. The convolution kernel of each channel has specific parameters so that CNNs can detect different features, which is the second property of convolution called channel-specific. Generally, the number of channels increases as a network deepens. Yet, it is questionable if more channels yield higher accuracy. When the quality of a training set is poor when models are complex, there may be overfitting risks that lead to poor performance. Hundreds of channels complicate a model and increase the overfitting risks. It has been shown that many convolution kernels are redundant in the channel dimension (Jaderberg, Vedaldi, & Zisserman, 2014), so the traditional convolution operation may increase computation yet fail to improve the model performance.

Considering the channel-specific property, which causes redundant channels in a model. Redundant channels complicate GANs, resulting in an overfitting phenomenon. The overly complex discriminator makes a model have more stringent evaluation

* Corresponding author.
E-mail addresses: 3220190462@bit.edu.cn (Y. Tian), xygong@jnu.edu.cn (X. Gong), 01068@bitzh.edu.cn (J. Tang), bhsu@263.net (B. Su), tlxx@jnu.edu.cn (X. Liu), Zhangxy@jnu.edu.cn (X. Zhang).

https://doi.org/10.1016/j.neunet.2022.05.014
0893-6080/

criteria for the generated false images. This will cause a severe problem for GANs, i.e., mode collapse, where the generator produces many similar images that lose diversity. In summary, how to appropriately improve the generated image quality without the overfitting risk is challenging.

Aiming to solve the problems of CNN-based GANs, we propose the GIU module for GANs to enhance the generated image quality. The GIU module comprises involution (Li et al., 2021) and the Squeeze-and-Extraction (SE) module (Hu, Shen, & Sun, 2018). Compared with traditional convolution, involution has the characteristics of Spatial-specific and channel-agnostic. It relies on the input feature map to generate kernel parameters and thus improves the feature extraction ability of the model with a small number of parameters. Meanwhile, due to its special kernel parameter generation method, we introduce the SE module to filter out the unfavorable features for the current task for the involution to enhance the feature extraction capability of the kernel. Compared with the traditional convolution, the GIU module has fewer parameters, and it can be well embedded in GANs to enhance the quality of the generated images and prevent overfitting.

In GAN-based image generation tasks, random noises are sampled to generate images by the generator, and BN (Ioffe & Szegedy, 2015) has become the core configuration of generators because of its stability training advantages. However, normalization causes generators to ignore the representation differences between different noises. To address this problem, we replace the BN in the generator with Representative Batch Normalization (RBN) (Gao, Han, Li, Cheng, & Peng, 2021). RBN introduces the formulation of centering and scaling calibrations to conventional BN. RBN compresses input features as representational features and enhances or suppresses them through learnable variables. Through RBN, GANs can enhance or suppress the representativeness difference of different random noises to enhance the generated image quality.

The contributions of this study can be summarized as follows.

- We design a new GIU module for GANs. The GIU module can exploit global information to improve the ability of GAN feature extraction and enhance the generated image quality.
- We introduce RBN to GANs for more focus on the expression of representative features.
- We adopt two technologies, spectral normalization (Miyato, Kataoka, Koyama, & Yoshida, 2018) and WGAN-GP (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017), to stabilize the training of GANs and improve the generated image quality.

We compared our model with many classical GAN models, such as Deep Convolutional GANs (DCGANs) (Radford et al., 2016) LSGANs (Mao et al., 2017), WGAN-GP, SNGANs (Miyato et al., 2018), and Self-Attention GANs (SAGANs) (Zhang, Goodfellow et al., 2019) on CIFAR-10 and CelebA datasets. Each GAN was trained 1 million times, and all models generated 50,000 images every 100,000 times. The images were tested to calculate Inception Score (IS) (Salimans et al., 2016) and Frechet Inception Distance (FID) (Liu, Luo, Wang, & Tang, 2015). For all models, we selected the one with the best scores in 10 tests as the best model.

The rest of this article is organized as follows. Section 2 reviews studies related to GANs. In Section 3, we introduce GIU-GANs, including the architecture of the GIU module and techniques for stabilizing the training of GANs. The algorithm of RBN is presented in Section 4. In Section 5, we present the experimental results of many classical GAN models and our model implemented on the CIFAR-10 and CelebA datasets. Finally, Section 6 concludes this study.

2. Related work

In this section, we review some previous studies related to GANs.

Image generation has long been a subject that has attracted researchers from a wide range of fields, including computer vision and machine learning. In 2014, GANs (Goodfellow et al., 2014) was proposed to generate images and has received extensive attention. Since then, various GANs have been developed. To enhance the generated image quality, Mirza and Osindero (2014) proposed conditional GANs, introducing the condition information as a label to the generator. In addition, Radford et al. (2016) proposed DCGANs to improve GANs by full convolutional structure. Further, interpretable representation learning by information maximizing GANs (Chen et al., 2016) constrained some dimensions of random noise to control the semantic characteristics of generated data. In addition, variational autoencoder with GAN (Larsen, Sønderby, Larochelle, & Winther, 2016) introduced a discriminator basis on the conventional variational autoencoder (Kingma & Welling, 2014) to better capture the data distribution. In addition, Energy-based GAN (Zhao, Mathieu, & LeCun, 2017) introduced an autoencoder to the discriminator, showing an improved convergence pattern and scalability to generate high-resolution images. While the theory of GANs has made rapid progress, the applications of GANs have also flourished. The applications include text-to-image tasks (Hong, Yang, Choi, & Lee, 2018; Reed et al., 2016; Zhang, Xu, & Li, 2017; Zhang et al., 2019), image-to-image tasks (Ioffe & Szegedy, 2015; Karras, Laine, & Aila, 2019; Nie, Narodytska, & Patel, 2019; Zhu, Park, Isola, & Efros, 2017), and image super-resolution tasks (Bulat, Yang, & Tzimiropoulos, 2018; Ledig et al., 2017; Wang et al., 2018).

Despite GANs having made significant progress, the training of GANs is still filled with difficulties. Poorly trained GANs may generate similar images and reduce the diversity of generated images, leading to mode collapse. To address the issue, multiagent diverse GANs (Ghosh et al., 2018) built multiple generators simultaneously, and each generator produced a different pattern to guarantee the diversity of samples produced. Moreover, improving objective function is a strategy to stabilize the training of GANs. Unrolled GANs (Metz, Poole, Pfau, & Sohl-Dickstein, 2017) changed the objective function of the generator considering both the current state of the generator and discriminator after K updates. Meanwhile, Mao et al. (2017) proposed LSGANs to stabilize the training of GANs. LSGANs changed the objective function of GANs from cross-entropy loss to least square loss and achieved a certain effect. Further, Wasserstein GANs (WGANs) (Arjovsky, Chintala, & Bottou, 2017) replaced Jensen–Shannon (JS) divergence with Wasserstein distance, which considerably stabilized the training of GANs. In addition, because WGANs used the weight clipping strategy to enforce the Lipschitz constraint, the value of weight will affect the training result. Therefore, Gulrajani et al. (2017) proposed WGAN-GP, which replaced the strategy of weight clipping based on WGANs with the gradient penalty to meet the Lipschitz constraint for the training of GANs. Moreover, to satisfy the Lipschitz constraint, SNGANs (Miyato et al., 2018) used spectral normalization to stabilize the training of GANs. On this basis, SAGANs (Zhang, Goodfellow et al., 2019) combined spectral normalization with self-attention (Vaswani et al., 2017) mechanism to improve the performance of GANs.

3. Global Information Utilization for Generative Adversarial Networks (GIU-GANs)

In this section, we present GIU-GANs in detail. Firstly, we introduce preliminary work on GIU-GANs, including SE Module

and involution. Then, the implementation method of the GIU module will be explained. Next, we show the architecture of GIU-GANs. Finally, we describe two techniques in detail to stabilize the training of GANs, WGAN-GP and spectral normalization for both generator and discriminator.

3.1. Preliminary work

In this subsection, we describe the channel attention SE module and involution in detail.

3.1.1. Squeeze-and-Excitation Module (SE module)

At present, CNNs are core components of Computer Vision research. Capturing more information from images is becoming increasingly relevant in CNNs. There has numerous study (Newell, Yang, & Deng, 2016) considering the spatial domain to improve the model performance, where as the SE module considers the relationship between characteristic channels. It captures the importance of each feature channel and then refers to this importance enhancement or suppression feature.

The first step is to compress each feature channel to a real number z using global average pooling (Lin, Chen, & Yan, 2014). Each real number obtained by compression has a global field of sensation, and their number of channels matches the number of input channels.

Then, the SE module captures the relationship between the channels via the excitation operation. It employs a gating mechanism in the form of the sigmoid function:

$$s = F_{ex}(z, w) = \sigma(g(z, W)) = \sigma(W_2\delta(W_1 z)), \tag{1}$$

where $W_1 \in R^{\frac{C}{r}\times C}$, $W_2 \in R^{C\times\frac{C}{r}}$. The SE module adopts full connection layer operation to reduce the dimension of s by squeeze, which can reduce the computational load. Then, the result of dimensionality reduction is processed using the Rectified Linear Unit (ReLU) activation function, which is then multiplied by the full connection layer W_2. The output dimension is $1\times1\times C$. Next, the feature maps are processed by the sigmoid function to obtain s.

Finally, the sigmoid activation value of each captured channel is multiplied by the original feature:

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c. \tag{2}$$

As such, useful feature channels are enhanced, and less useful feature channels are suppressed.

3.1.2. Involution

Involution, which is a new neural network operator, has different characteristics from convolution. Involution has the characteristics of spatial-specificity and channel-agnostic. The former property contributes to generating parameters of the kernel by feature maps to improve the ability of feature extraction. The latter property can reduce the number of channels, thereby significantly decreasing the computational load and preventing model overfitting. The involution kernel size is $H \times W \times K \times K \times G$, where K is the size of the involution kernel, we set it to 3 in this study (equivalent to the conventional 3×3 convolution) H and W are resolutions of the feature map. G represents the number of groups. Thus, the feature map is divided into G groups, with $\frac{C_i}{G}$ channels in each group ($G \ll C_i$). Involution is defined as follows:

$$Y_{i,j,k} = \sum_{(u,v)\in\Delta k} H_{i,j,u+[K/2],v+[K/2],[kG/C]} X_{i+u,j+v,k}, \tag{3}$$

where $H \in R^{H\times W\times K\times K\times G}$ is the approach by which the involution kernel is created:

$$H_{i,j} = \phi(X_{\Psi_{i,j}}). \tag{4}$$

$\Psi_{i,j}$ is the index set in the neighborhood of pixel (i, j) to obtain parameters of number $1 \times 1 \times C$, whereas ϕ is a series of scaling and reshape operations. Selecting a single point set of i, j on the feature map can acquire involution kernel instantiation:

$$H_{i,j} = \phi(X_{i,j}) = W_1\sigma(W_0 X_{i,j}), \tag{5}$$

where $W_0 \in R^{\frac{C}{r}\times C}$, $W_1 \in R^{K\times K\times G\times\frac{C}{r}}$. Involution uses W_0 and W_1 to transform the parameter matrix to $K \times K \times G$; r is the channel reduction ratio, and σ is the BN and ReLU operations.

Involution turns the feature vector of pixel (i, j) in a feature map into $K \times K \times G$ by ϕ and reshapes the operation, where $K \times K$ is the kernel shape. Finally, involution performs multiplication–addition operations on the weight matrix of size $K \times K \times G$ and the feature vector of the neighborhood of pixel to obtain the final outputting feature map. Involution divides the channels of an input feature map into G groups, with $\frac{C_i}{G}$ channels in a group, and extracts a set of points in the neighborhood $K \times K$ size of a certain pixel within the group for multiplication–addition operation with the generated involution kernel. Therefore, the point set of a pixel in a group and the involution kernel multiplication–addition does not change the number of output channels. For other feature extraction operations that do not change the number of output channels, such as 1×1 convolution or 3×3 convolution, the number of output channels needs to be the same as the number of input channels, i.e., $C_i = C_o$. However, the G of involution is much smaller than the input channels. Using a smaller G instead of C_o can effectively avoid the generation of redundant channels and thus prevent overfitting. Meanwhile, the spatial-specificity of involution makes its kernel generated on a per-pixel basis on the feature map, thereby preventing it from producing the same response in a single feature map, i.e., all its kernels are not the same. Therefore, unlike conventional convolution, involution is more concerned with the diversity of different regions in a feature map, which is reflected in GANs by its ability to extract detailed features. Section 5 confirms the superiority of GIU-GANs in terms of detailed features.

3.2. Global Information Utilization (GIU) module

In this subsection, we present the algorithm of the GIU module comprising the SE module and involution.

3.2.1. Algorithm of GIU module

Notably, the involution kernel generative function ϕ considers using the feature vector of point (i, j) on the feature map for design:

$$\phi(X_{i,j}) = W_1\sigma(W_0 X_{i,j}), \tag{6}$$

where $X_{i,j}$ is the feature vector of the pixel located in point (i, j). The value of the feature vector represents the output of point (i, j) on each channel. Different channel responses have different effects on the current task; useful values will improve the performance of GANs, whereas less useful values will misguide GANs. To obtain a more effective involution kernel, we combine the SE module with involution, called the GIU module, to adaptively enhance kernel useful parameters and suppress useless parameters. Fig. 1 is a schematic of the architecture of the GIU module.

First, the SE module is employed to capture the importance of each channel for the input feature map:

$$s = \sigma(W_2\delta(W_1 F_{sq}(X))), \tag{7}$$

where $W_1 \in R^{\frac{C}{r}\times C}$, $W_2 \in R^{C\times\frac{C}{r}}$. $F_{GAP}(X)$ is global average pooling operation:

$$z_c = F_{GAP}(X) = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{i=1}^{W} u_c(i, j), \tag{8}$$

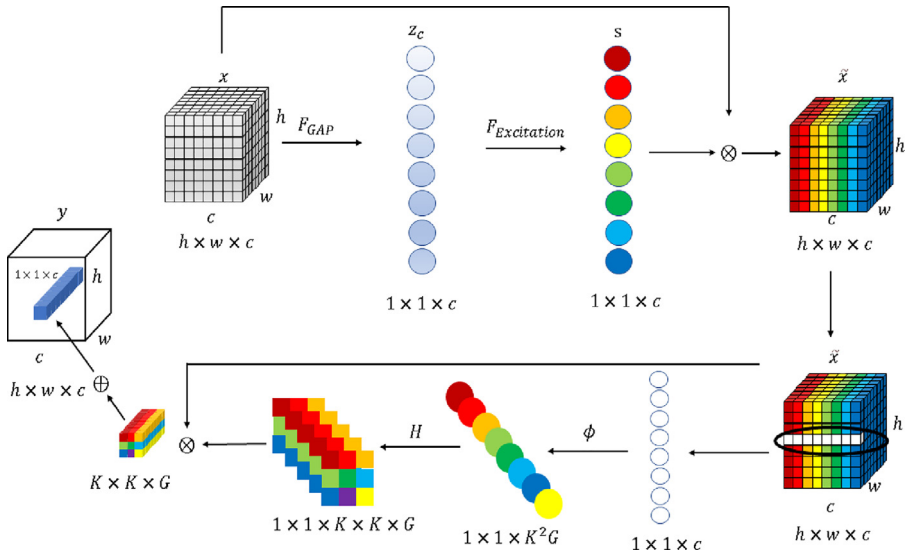


Fig. 1. A schematic of the GIU module for GIU-GANs. The ⊗ and ⊕ represent matrix multiplication and addition, respectively.

u_c are feature maps where input X uses a convolution operation, H and W are spatial dimensions of feature maps, and s is the importance of each channel. Multiplying the s of each channel by itself, we have the following: $\tilde{X} = s \cdot u_c$, where $u \in R^{H \times W \times C}$ is a two-dimensional spatial kernel and u_c represents the cth channel.

Then, the obtained $\tilde{X}$ is used to generate the involution kernel:

$$\tilde{H}_{i,j} = \phi(\tilde{X}_{\psi_{i,j}}). \tag{9}$$

The previous involution kernel is the point set at pixel (i, j): $H_{i,j} = [x_c^1, x_c^2, \ldots, x_c^{K^2}]$, and the improved kernel function is $\tilde{H}_{i,j} = [\tilde{x}_c^1, \tilde{x}_c^2, \ldots, \tilde{x}_c^{K^2}]$. The new generation function can select channels, thereby adaptively adjusting the parameter matrix according to the importance of the channel. Reshaping the involution kernel size to $K \times K \times G$, where G is the number of groups with each group sharing the same involution kernel, then the final feature map can be obtained by multiplication–addition using the feature vector of the GIU module and the neighborhood of the corresponding point on the input feature map:

$$\tilde{Y}_{i,j,k} = \sum_{(u,v) \in \Delta k} \tilde{H}_{i,j,u+[K/2],v+[K/2],[kG/C]} X_{i+u,j+v,k}. \tag{10}$$

Specifically, the algorithm of the GIU module is shown in Algorithm 1.

GIU-GANs introduces the GIU module, which combined the channel attention mechanism SE module and involution together. The SE module uses two operations, i.e., squeeze and excitation, to explicitly model the interdependence between feature channels. The SE module uses the relationship between channels to obtain the importance degree of each channel through learning. It enhances the useful features according to this importance degree and prevents unuseful features for the current task. From the perspective of the number of parameters, the number of parameters in the GIU module is $\frac{(C_i^2 + K^2 G C_i)}{r_1} + 2C_i + \frac{(2C_i^2)}{r_2}$, where $\frac{(C_i^2 + K^2 G C_i)}{r_1}$ is the number of involution parameters, $2C_i$ is the number of Batch Normalization (BN) parameters, $\frac{(2C_i^2)}{r_2}$ is the number of SE module parameters, r_1 is the ratio of channel reduction, which we set to 16, and r_2 is the channel scaling ratio for the SE module (we follow the original SE module (Hu et al., 2018) paper and set it to 16). Relative to the number of convolution parameters $k^2 C_i^2 C_o^2$, C_o is the number of output channels, which is often set to hundreds or thousands of numbers, and we set G much smaller than C_o. In summary, the number of parameters of the GIU module is much smaller than that of conventional convolution. We counted the number of parameters for all evaluated models and added them. According to the GIU module's principle and parameter formula, it is suitable to be inserted after the "small feature map with more channels". The experiment in Section 5.1 helps us verify this view.

Algorithm 1 Algorithm of GIU module

Input: The set of feature maps for current batch, X_c, its spatial dimensions are $H \times W$.
$W_0 \in R^{\frac{C}{r} \times C}$, $W_1 \in R^{C \times \frac{C}{r}}$ are full connection layer operations.
$W_2 \in R^{\frac{C}{r} \times C}$, $W_3 \in R^{K \times K \times G \times \frac{C}{r}}$ are convolution operations.
σ is sigmoid activation function, δ is ReLU activation function.
Output: The set of feature maps for current batch, $\tilde{Y}_{i,j,k}$

$z_c \leftarrow \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{i=1}^{W} X_c(i, j)$.
$s \leftarrow \sigma(W_1 \delta(W_0 z_c))$.
$\tilde{X}_c \leftarrow s \cdot u_c$.
$\tilde{H}_{i,j} \leftarrow W_3 \delta(W_2 \tilde{X}_c)$.
Reshaping the size of involution kernel: $1 \times 1 \times K^2 G \xrightarrow{reshape} K \times K \times G$.
$\tilde{Y}_{i,j,k} \leftarrow \sum_{(u,v) \in \Delta k} \tilde{H}_{i,j,u+[K/2],v+[K/2],[kG/C]} X_{i+u,j+v,k}$.
return $\tilde{Y}_{i,j,k}$.

3.3. The architecture of GIU-GANs

The architecture of GIU-GANs is shown in Figs. 2 and 3.

The discriminator does not use BN, and GIU-GANs select LeakReLU with the slope of the leak set to 0.1 and ReLU for the discriminator and generator, respectively.

The GIU module focuses on the importance of the channels in the hidden layer and uses this as an important basis for generating the parameter matrix. Meanwhile, the formula of the number of parameters of the GIU module is expressed as follows:

$$\frac{C_i^2 + K^2 G C_i + r_i C_i}{r_i} + \frac{2C_i^2}{r_2}. \tag{11}$$

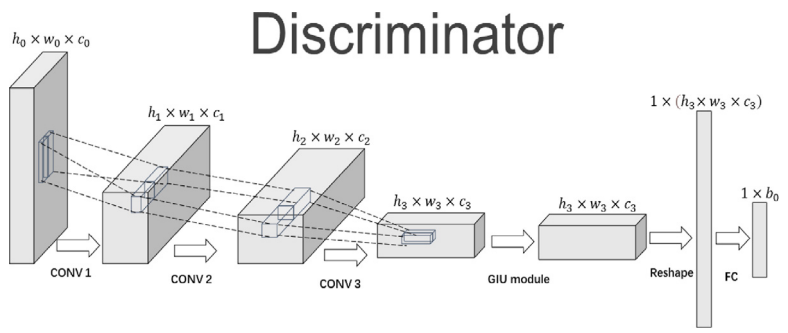


Fig. 2. The discriminator architecture of GIU-GANs. h_x and w_x, respectively, denote the height and width of the current feature map, c_x denotes the number of channels in the current hidden layer, and b_0 denotes the batch size.

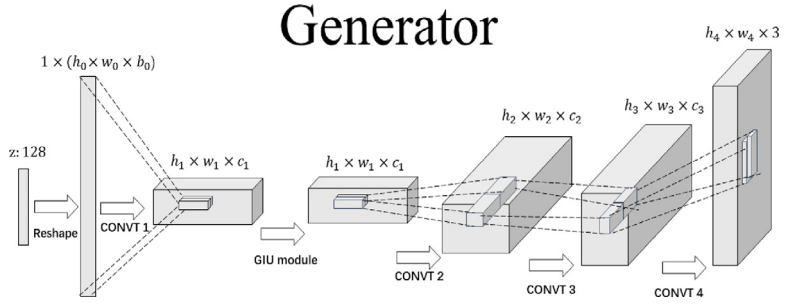


Fig. 3. The generator architecture of GIU-GANs. h_x and w_x, respectively, denote the height and width of the current feature map, c_x denotes the number of channels in the current hidden layer, and b_0 denotes the batch size.

Considering the necessity of reducing the number of model parameters, we insert the GIU module into the latter layer of the input layer of the generator and the former layer of the output layer of the discriminator. The experiments in Section 5 demonstrate the validity of our scheme.

3.4. Two technologies for training GANs

3.4.1. Spectral normalization for both generator and discriminator

As we know, WGAN replaced the JS divergence with the Wasserstein divergence. However, this requires the discriminator to satisfy the Lipschitz constraint:

$$|f(x_1) - f(x_2) \leq K|x_1 - x_2|. \tag{12}$$

It requires that the absolute value of the derivative of the function f does not exceed K. Otherwise, gradient explosion will occur during model training. SNGANs satisfy the Lipschitz constraint using spectral normalization by employing the spectral normalization of each layer $\sigma(A)$ to restrict the parameter matrix W_i:

$$\bar{W}_i = \frac{W_i}{\sigma(A)}, \tag{13}$$

where $\bar{W}_i$ is the parameter matrix after spectral normalization. Spectral normalization has the advantage of not requiring extra hyperparameter tuning. In addition, the computational cost is relatively less. In conclusion, it can be used to satisfy the Lipschitz constraint and stabilize the training of GANs.

In GIU-GANs, we borrowed from SAGANs: spectrum normalization is applied to both discriminator and generator to stable the training of stable GANs.

3.4.2. WGAN-GP

Moreover, we adopt WGAN-GP as our objective function. WGAN-GP uses a new truncation strategy, gradient penalty::

$$L = E_{\tilde{x}\sim P_g}[D(\tilde{x})] - E_{\tilde{x}\sim P_r}[D(\tilde{x})] + \lambda E_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{14}$$

where P_r is the real image distribution, P_g is the generated image distribution, $P_{\hat{x}}$ is the distribution interpolated between P_r and P_g, and λ is the penalty coefficient. The gradient penalty, $|\nabla_{\hat{x}} D(\hat{x})| \leq K$, satisfies the Lipschitz constraint to stabilize training of GANs. Specifically, the training details of GIU-GANs are summarized in Algorithm 2:

4. The algorithm of Representative Batch Normalization (RBN)

The conventional BN performs three operations on input feature maps X: centering, scaling, and affine operations, respectively, given by

$$\begin{aligned} &\text{Centering}: X_m = X - E(X), \\ &\text{Scaling}: X_s = \frac{X_m}{\sqrt{Var(X) + \epsilon}}, \\ &\text{Affine}: Y = X_s\gamma + \beta. \end{aligned} \tag{15}$$

where $E(X)$ and $Var(X)$ represent the mean and variance of feature maps, respectively, γ and β denote the learnable scaling and offset terms, respectively, in the affine transformation, and ϵ is used to prevent the occurrence of zero values. In the model training process, the mean and variance are calculated in a mini-batch:

Algorithm 2 Training Algorithm of GIU-GANs. We use default values of $\lambda = 10$, $\alpha = 0.0002$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Input: The batch size m, Adam hyperparameters α, β_1, β_2, the gradient penalty coefficient λ.

Input: Initial discriminator parameters w_0, initial generator parameters θ_0

for number of training iteration **do**
  for i = 1,..., m **do**
    Sample real data $x \sim P_r$, latent variable $z \sim p_(z)$, a random number $\epsilon \sim U[0, 1]$.
    $\tilde{x} \leftarrow G_{\theta_G}(z)$.
    $\tilde{x} \leftarrow \epsilon x + (1 - \epsilon)\hat{x}$.
    $L^{(i)} \leftarrow D_w(\tilde{x}) - D_w(x) + \lambda(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2$.
  $w \leftarrow Adam(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$.
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$
  $\theta \leftarrow Adam(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$
return θ, w

$$\mu_B = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{n,c,h,w},$$
$$\sigma_B^2 = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (X_{n,c,h,w} - \mu_B)^2. \tag{16}$$

However, $E(X)$ and $Var(X)$ of mini-batch use normalization to reduce the expression of partial random noise by the generator. Therefore, we replace the BN with RBN in part of the network layer of GANs.

First, RBN introduces a centering calibration operation before the centering operation of BN:

$$X_{cm(n,c,h,w)} = X_{n,c,h,w} + w_m \bigodot K_m, \tag{17}$$

where $w_m \in R^{1\times C\times 1\times 1}$ is the learnable variable and K_m is the statistical features obtained after a series of processing of X. Different operations will result in different shapes of statistical features. For example, $K_m \in R^{N\times C\times 1\times 1}$ is to obtain statistical information of each channel through feature map normalization, or $K_m \in R^{N\times 1\times H\times W}$ is to obtain statistical information of each pixel after channel normalization. In GIU-GANs we follow the original RBN paper (Gao et al., 2021) and adopt global average pooling to normalize the feature map as the shape of $K_m \in R^{N\times C\times 1\times 1}$. The new mean is $E(X_{cm}) = (1 + w_m) \cdot E(X)$, it subtracts the old mean as follows:

$$\begin{aligned}&(X_{cm} - E(X_{cm})) - (X - E(X))\\ &= X + w_m \cdot K_m - ((1 + w_m) \cdot E(X) - (X - E(X)))\\ &= w_m \cdot (K_m - E(X)).\end{aligned} \tag{18}$$

After the centering calibration, w_m, as a learnable variable, will enhance or suppress representational noise. If $w_m > 0$ when $K_m > E(X)$, representative noise characteristics will be enhanced, whereas if $w_m < 0$ when $K_m > E(X)$, representative noise features will be suppressed.

Meanwhile, RBN adds a scaling calibration after the original scaling operation:

$$X_{cs(n,c,h,w)} = X_{s(n,c,h,w)} \cdot R(w_v \bigodot K_s + w_b), \tag{19}$$

where $w_v, w_b \in R^{1\times C\times 1\times 1}$ are learnable variables, which control the strength and position of the limiting function, respectively. Because $0 < R() < 1$, there must be a value τ that is $R() < \tau < 1$. Therefore, $Var(X_{cs})$ is expressed as follows:

$$Var_{X_{cs}} < Var(X_s \tau) = \tau^2 Var(X_s). \tag{20}$$

After the scaling calibration, the dispersion of the characteristic variance is reduced and a more stable channel distribution is obtained.

In Section 5, we compared GIU-GANs with and without RBN on the CIFAR-10 and CelebA datasets. The experimental results revealed the effectiveness of GIU-GANs with RBN on the CelebA dataset.

5. Experimental results

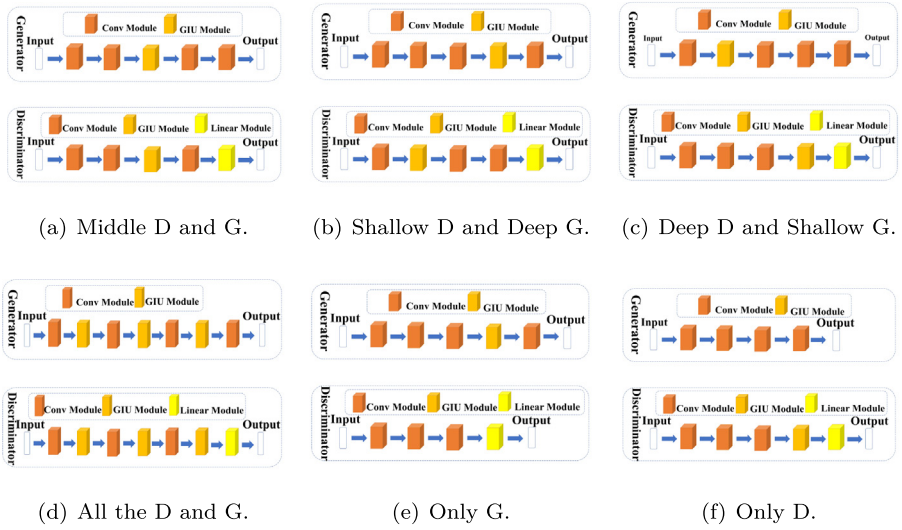
To evaluate the effectiveness of GIU-GANs, we performed experiments on two datasets: CelebA and CIFAR-10. The CIFAR-10 dataset comprises 50,000 and 10,000 training and testing images, respectively. We trained with all 50,000 images of the CIFAR-10 dataset. For the CelebA dataset, we aligned and cropped for 200,000 images to 64×64 resolution and used them as the training set. Moreover, to further verify the effectiveness of the GIU module and RBN, we added an ablation experiment in Section 5.5. Two common evaluation metrics, IS and FID, were used as the main metrics. IS employs pretrained InceptionNet-V3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) to evaluate generated images. The output of InceptionNet-V3 is a 1000-dimensional vector. The values of each dimension of the output vector correspond to the probability that the picture belongs to a certain category. For IS, it needs to consider the quality of a single image and the diversity of a batch of images. For a single image, the entropy of the probability distribution of InceptionNet-V3 output should be minimal. Smaller means that the generated image is more likely to belong to a certain category and the image quality is high. For a batch of images generated, the average probability distribution entropy of InceptionNet-V3 output should be maximal. In other words, to ensure the diversity of images generated, GANs needs to make the output of InceptionNet-V3 spread equally across 1000-dimensional labels. The formula for IS is expressed as follows:

$$IS = exp(E_{x\in p_g} D_{KL}(P(y|x) \parallel P(y))), \tag{21}$$

where $P(y|x)$ denotes the output distribution of the generated image x input to InceptionNet-V3, $P(y)$ represents the average distribution of generated images in the InceptionNet-V3 output category, and D_{KL} means KL divergence. In summary, IS is measured for a single dataset only. To measure the probability distribution of a dataset, mean and variance are crucial, so standard deviation needs to be added to determine the distribution of this dataset when calculating IS. FID measures the generated image quality by comparing the "distance" of the generated image to the real image; the lower the value the better. FID does not use the last layer of InceptioNet-V3 as the output, but the last pooled layer as the output. This layer will output a vector of 2048 dimensions. This makes each image is predicted to have 2048 features. Next, FID calculates the mean of the 2048-dimensional eigenvectors and compares them with the covariance matrix. Thus, the FID score is defined as follows:

$$FID(x, g) = \|\mu_x - \mu_g\|_2^2 + Tr(\sum\nolimits_x + \sum\nolimits_g - 2(\sum\nolimits_x \sum\nolimits_g)^{\frac{1}{2}}), \tag{22}$$

where μ_x and μ_g are the means of two distributions; $\sum_x$ and $\sum_g$ are the covariance of the two distributions, respectively; Tr is the trace operation of the matrix, i.e., the sum of all elements of the main diagonal of the matrix. FID is used to calculate the distance between two distributions; a lower FID means a closer



(a) Middle D and G. (b) Shallow D and Deep G. (c) Deep D and Shallow G.

(d) All the D and G. (e) Only G. (f) Only D.

Fig. 4. GIU module in GANs multiple insertion situations.

Table 1
Experimental results of different locations and numbers of GIU modules on the CIFAR-10 dataset.

Model	FLOPs	Params	IS(↑)	FID(↓)
Middle D and G	97.24m	1.86M	4.36 ± 0.05	26.72
Shallow D and Deep G	98.15M	1.92M	3.96 ± 0.04	39.53
Deep D and Shallow G	97.23M	1.92M	4.72 ± 0.06	22.66
All the D and G	99.03M	1.94M	4.21 ± 0.04	35.36
Only G	96.79M	1.88M	4.52 ± 0.06	26.24
Only D	96.79M	1.88M	4.44 ± 0.07	29.84

match between the generated image distribution and the real image distribution. It considers the mean and variance while calculating the two distributions, so there is no need to add standard deviation when calculating an FID score.

All models were trained 1 million times, and 50,000 images generated by all models were measured by PyTorch implementations of IS and FID. All experiments were set with NVIDIA RTX 3090 GPU and Intel Core i9-9900k CPU@3.60 GHz using PyTorch 1.7.0. In the following, we present experimental results of GIU-GANs and compare them with other GANs.

5.1. Hyperparameters study

To discuss how to insert the GIU module for the best performance of GANs, we trained different GIU-GANs on CIFAR-10; their IS and FID are summarized in Table 1. In the following table, we use "D" for discriminator, "G" for generator, "shallow" for the shallow layer of the network, and "deep" for the deep layer of the network. Fig. 4 shows the insertion of different GIU modules.

After identifying the location of the GIU module, to verify which hyperparameters are suitable for our model, different hyperparameters (optimization algorithm, learning rate, Different K, and different grouping numbers) were applied to GIU-GANs and trained on the CIFAR-10 dataset. We list the details of these hyperparameters in Tables 2 and 4 and record the experimental results in Tables 3 and 4.

As can be seen from the IS and FID of the model participating in the experiment, GIU-GANs-V3 should be selected for setting the hyperparameters of GIU-GANs.

Meanwhile, different K implies that the involution kernel has different sizes, and different G represents different numbers of channels sharing the kernel. In our experiments, we use the hyperparameter *Group_Channelss* to set the $G = \frac{C_i}{Group\_Channels}$. From the parameter formula of the GIU module $\frac{(C_i^2 + K^2 G C_i)}{r_1} + 2C_i + \frac{(2C_i^2)}{r_2}$, we know that different K and G indicate different numbers of parameters; to further explore their effects on GIU-GANs, we set up GIU-GANs with different K and *Group_Channels* on the CIFAR-10 dataset for an experiment. The results are summarized in Table 4.

Through experimental comparison, Adam was set as the optimization algorithm of GIU-GANs; the generator and discriminator's learning rates were set to 0.0002; K should be set to 3, and *Group_Channels* should be 16.

5.2. Results on the CIFAR-10 dataset

The architecture of the GIU-GANs used to train the CIFAR-10 dataset is shown in Fig. 5:

In this experiment, GIU-GANs and other counterpart GANs were all trained 1 million times on the CIFAR-10 dataset. All GANs generated 32×32 images, and we calculated their IS and FID scores. The best results of all models are shown in Table 5.

Fig. 6 shows images generated by all participating GANs; from Table 5, GIU-GANs outperformed other well-known GANs.

5.3. Results on the CelebA dataset

The architecture of the GIU-GANs used to train the CelebA dataset is shown in Fig. 7:

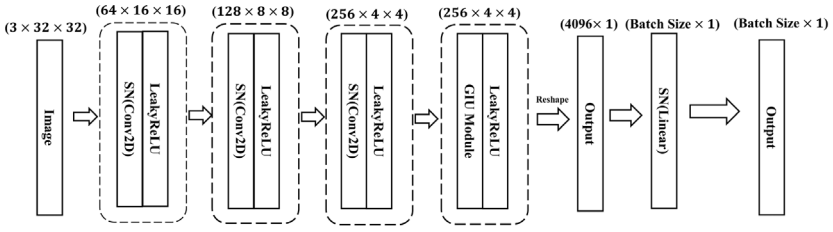
In this experiment, we cropped and aligned all images from the CelebA dataset for training and trained 1 million times. All GANs generated 50k 64×64 images, and we calculated their IS and FID scores.

In addition, as shown in Fig. 8, when DCGANs were trained on the CelebA dataset, the gradient-vanishing of the discriminator resulted in no change of generator gradient. Thus, we no longer used DCGANs for comparison.

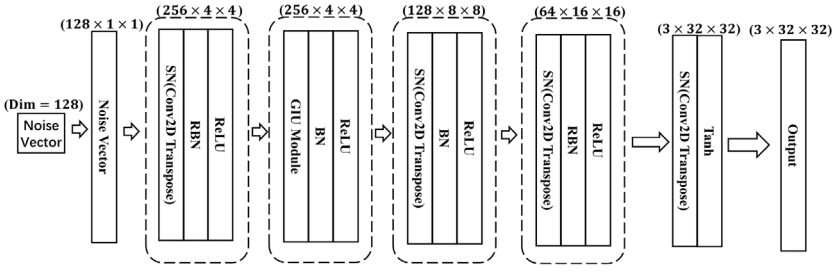
The best results of all models are shown in Table 6 and the images generated by GIU-GANs on the CelebA dataset are shown in Fig. 9. From Table 6, GIU-GANs performed best in both IS and FID.

Table 2
Table of different GIU-GANs parameter settings.

Model	D_{lr}	G_{lr}	Batch Size	Optimization Algorithm	FLOPs	Params
GIU-GANs-V1	0.0001	0.0004	64	Adam (Kingma & Ba, 2015)	97.23M	1.92M
GIU-GANs-V2	0.0003	0.0003	64	Adam	97.23M	1.92M
GIU-GANs-V3	0.0002	0.0002	64	Adam	97.23M	1.92M
GIU-GANs-V4	0.0002	0.0002	64	Adagrad (Duchi, Hazan, & Singer, 2011)	97.23M	1.92M
GIU-GANs-V5	0.0002	0.0002	64	RMSprop	97.23M	1.92M



(a) Discriminator Network.



(b) Generator Network.

Fig. 5. The architecture of GIU-GANs applied on CIFAR-10.

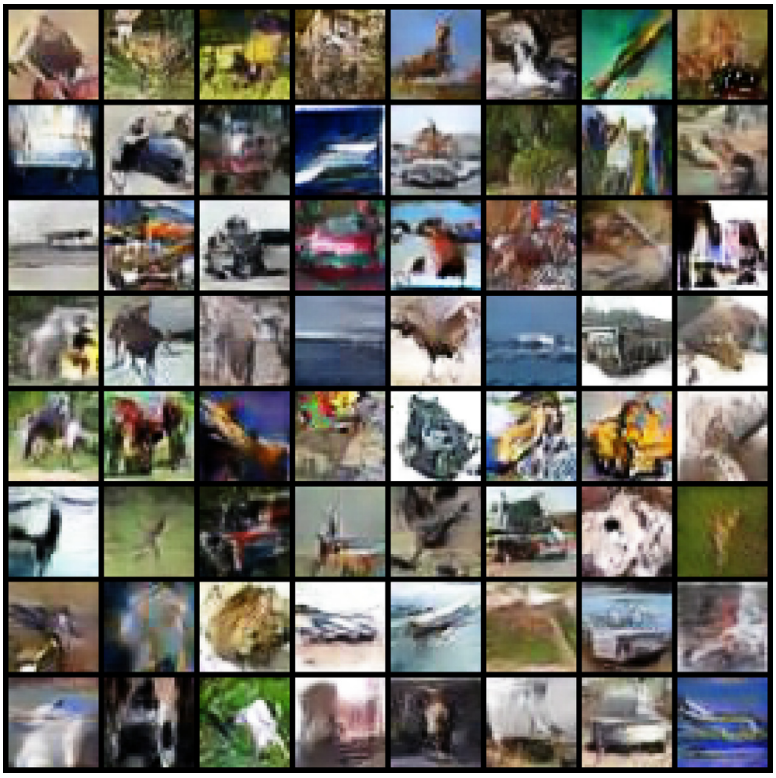
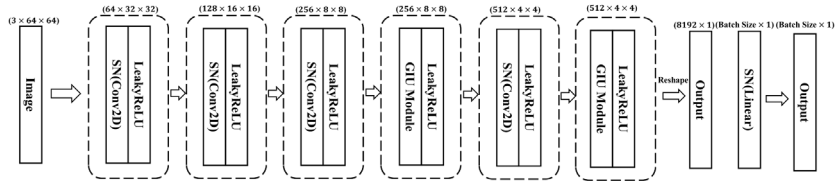
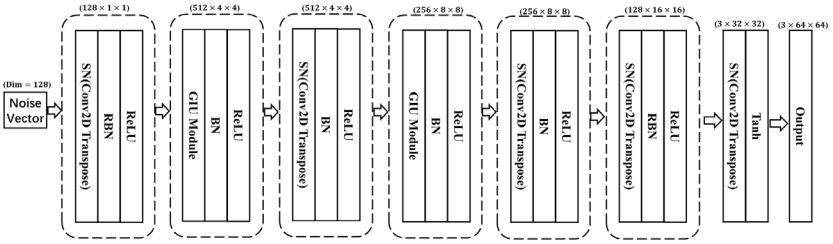


Fig. 6. Examples randomly produced by GAN models on CIFAR-10.

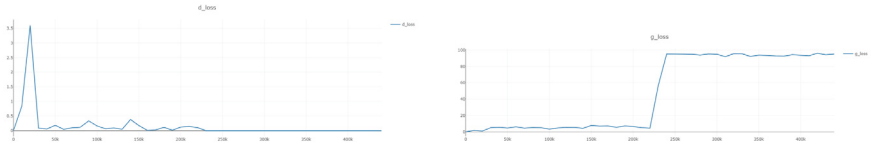


(a) Discriminator Network.



(b) Generator Network.

Fig. 7. The architecture of GIU-GANs applied on CelebA.



(a) Convergence curve of the discriminator of DCGANs on CelebA.

(b) Convergence curve of the generator of DCGANs on CelebA.

Fig. 8. Convergence curve of DCGANs on CelebA.

Fig. 9. Examples randomly produced by GAN models on CelebA.

Table 3
Experimental results of GIU-GANs with different parameter settings on CIFAR-10.

Model	FLOPs	Params	IS(↑)	FID(↓)
GIU-GANs-V1	97.23M	1.92M	4.39 ± 0.12	34.80
GIU-GANs-V2	97.23M	1.92M	4.78 ± 0.06	23.18
GIU-GANs-V3	97.23M	1.92M	4.72 ± 0.06	22.66
GIU-GANs-V4	97.23M	1.92M	3.25 ± 0.03	61.28
GIU-GANs-V5	97.23M	1.92M	4.47 ± 0.06	28.93

Table 4
Effects of different K and *Group_Channels* for GIU-GANs on CIFAR-10.

K	*Group_Channels*	Params	FLOPs	IS(↑)	FID(↓)
3	16	97.23M	1.92M	4.72 ± 0.06	22.66
5	16	97.76M	1.95M	4.64 ± 0.06	24.71
7	16	98.56M	1.99M	4.73 ± 0.03	22.97
3	8	97.52M	1.93M	4.58 ± 0.06	24.50
3	32	97.08M	1.91M	4.65 ± 0.06	24.89

Table 5
IS and FID scores of DCGANs, WGAN-GP, LSGANs, SNGANs, SNGANs and GIU-GANs on CIFAR-10.

Model	FLOPs	Params	IS(↑)	FID(↓)
DCGANs	309.24M	4.34M	4.34 ± 0.05	26.22
WGAN-GP	96.39M	1.85M	4.51 ± 0.06	22.95
LSGANs	117.25M	1.15M	3.24 ± 0.03	66.51
SNGANs	96.36M	1.85M	4.40 ± 0.04	24.60
SAGANs	101.65M	1.98M	3.52 ± 0.03	67.31
GIU-GANs	**97.23M**	**1.92M**	$\mathbf{4.72 \pm 0.06}$	**22.66**

Table 6
IS and FID scores of WGAN-GP, LSGANs, SNGANs, SNGANs and GIU-GANs on CelebA.

Model	FLOPs	Params	IS(↑)	FID(↓)
WGAN-GP	536.58M	6.57M	2.48 ± 0.03	11.07
WGAN-GP(+2)	687.72M	12.47M	2.50 ± 0.02	12.39
WGAN-GP(+3)	764.48M	12.66M	2.57 ± 0.02	8.98
LSGANs	694.48M	1.89M	2.40 ± 0.02	15.48
SNGANs	536.48M	6.57M	2.54 ± 0.02	7.09
SAGANs	557.58M	7.00M	2.11 ± 0.02	19.10
GIU-GANs	**543.28M**	**6.91M**	$\mathbf{2.61 \pm 0.02}$	**6.34**

Table 7
Experimental results of GIU module ablation.

Model	FLOPs	Params	IS(↑)	FID(↓)
baseline	96.36M	1.85M	3.83 ± 0.05	37.54
GIU-GANs without GIU module	96.36M	1.85M	3.99 ± 0.05	39.97
GIU-GANs	97.23M	1.92M	$\mathbf{4.72 \pm 0.02}$	**22.66**

5.4. GANs visualization exploration

To explore the role of the GIU module in the discriminator, we employ principal component analysis to visualize the hidden layers of the discriminator. PCA can reduce high-dimensional M data to low-dimensional data $N(M > N)$ and retain the previous information as much as possible, Since the convolutional feature maps often have hundreds of channels, using PCA can effectively visualize the hidden layers of neural networks. First, we subtract the mean of each dimensional feature of the input feature map X and calculate its covariance matrix $\frac{1}{n}XX^T$, where n denotes the number of samples. Second, we employ the eigenvalue decomposition method to find the eigenvalues and eigenvectors of the covariance matrix $\frac{1}{n}XX^T$. Then, we sort the eigenvalues in descending order and select the largest D eigenvalues among them. To better observe the extraction of details, we choose $D = 1$, i.e., to generate a gray-scale map to observe the results of its feature capture. Next, D feature vectors are used as row vectors to form a feature vector matrix P. Finally, data are transformed into a new space constructed by the D feature vectors. Fig. 10 shows the hidden layers of discriminators for Least Squares GANs (LSGANs) (Mao et al., 2017), Wasserstein GAN with Gradient Penalty (WGAN-GP) (Gulrajani et al., 2017), Spectrally Normalized GANs (SNGANs) (Miyato et al., 2018), and GIU-GANs (our proposed model) on the CelebA dataset.

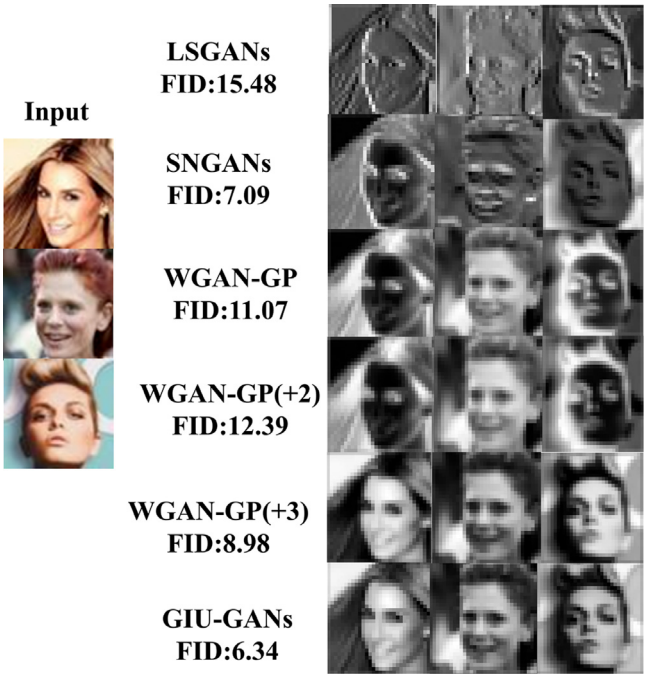


Fig. 10. Visualization of discriminator hidden layer for LSGANs, WGAN-GP, SNGANs, and GIU-GANs on CelebA.

For convolution, deeper networks can make the task of the convolutional kernel of each layer more explicit and thus effectively help the model fit the features better. For example, when the network is deep, the first layer learns the edge features, the second layer learns the simple shapes, and the third layer learns the target's complex shapes. However, when the network is shallow, the first layer is hard to learn the target's complex shapes. Therefore, deeper models can have stronger feature extraction ability. Motivated by this, we train several deeper WGAN-GP on the CelebA dataset and present the results in Fig. 10 to observe the extraction of detailed features. +2 and +3 mean that two and three layers, respectively, have been added to both the WGAN-GP discriminator and generator. Fig. 10 and Table 5 show that the WGAN-GP with three extra layers (parameters: 12.66M) achieved results comparable to those of the GIU-GANs (parameters: 6.91M). Through the GIU module, we could improve the feature extraction capability with fewer parameters for GANs and thus enhance the generated image quality.

After the neural network visualization, it can be found that the hidden layer of GIU-GANs has the best visual effect, and the image after feature extraction retains better-detailed features, which directly affects the score of the evaluation index FID.

Meanwhile, in order to better observe the quality of the generated image, we adopt the Bicubic method to upsample the generated images from 64×64 to 256×256, as shown in Fig. 11. Bicubic leverages the gray values of the 16 points around the point to be sampled for cubic interpolation. The influence of the gray level of the directly adjacent points, and the influence of the gray value change rate between the adjacent points is considered. After processing by this algorithm, a high-resolution image with a better visual effect can be obtained.

Table 8
Experimental results of RBN ablation.

Method	CIFAR-10				CelebA			
	FLOPs	Params	IS(↑)	FID(↓)	FLOPs	Params	IS(↑)	FID(↓)
GIU-GANs without RBN	97.22M	1.92M	4.79 ± 0.03	22.12	543.28M	6.91M	2.50 ± 0.03	6.53
GIU-GANs	97.23M	1.92M	$\mathbf{4.72 \pm 0.06}$	**22.66**	543.28M	6.91M	$\mathbf{2.61 \pm 0.02}$	**6.34**

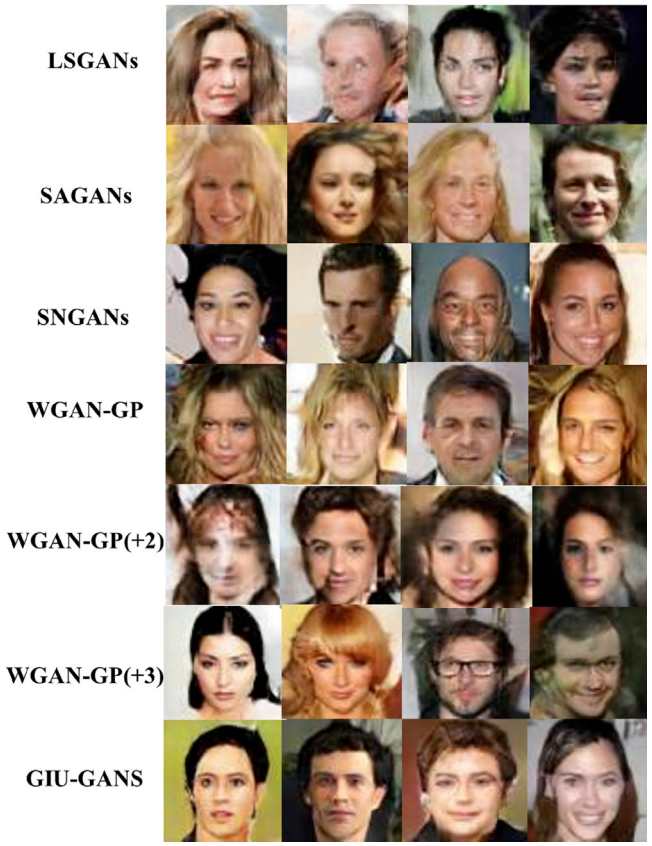


Fig. 11. Examples randomly produced by GAN models on CelebA and all generated images were upsampled by a factor of 4 using the Bicubic method.

From Fig. 10 we can observe the ability of the discriminator of GIU-GANs to capture details, and Fig. 11 shows that the generator of GIU-GANs has greater advantages in generating details. These advantages are also directly reflected in the visual quality of the generated images, IS, and FID.

5.5. Ablation study

To explore the reasons for the superior performance of GIU-GANs, several ablation studies were designed to study the contributions of individual components. We applied the GIU-GANs with and without the GIU module on the CIFAR-10 dataset. In addition, we employed WGAN-GP with spectral normalization for the generator and discriminator as the baseline. The best results are shown in Table 7.

From Table 7, the performance of GIU-GANs deteriorated significantly when the GIU module was removed.

Table 8 shows the influence of RBN on GIU-GANs. GIU-GANs without RBN on the CIFAR-10 dataset performed well; however, for the CelebA dataset, GIU-GANs with RBN was superior.

5.6. Results analysis

The training of GANs is extremely unstable. After a long time of training, the error increases, resulting in the overfitting phenomenon. In GANs, some hidden layers have thousands of channels, and redundant channel information not only increases the computational time of the model but also makes the model more complex and increases the overfitting risk. The GIU module is placed after the hidden layer with a large number of channels. The GIU module first filters the global information, enhances useful information, and suppresses useless information. Then, the GIU module pays attention to each pixel of the feature maps and generates the corresponding parameter matrix for multiplication–addition. By filtering the global channels and paying attention to each pixel on the feature maps, the GIU module can exploit global information. The performance of GIU-GANs can be improved without stacking convolutions, so GIU-GANs can easily enhance the generated image quality without much consideration of the overfitting risk.

Moreover, WGAN-GP achieved results similar to GIU-GANs on the CIFAR-10 dataset. We argued that the computational complexity of WGAN-GP is FLoating-point OPerations (FLOPs): 96.39M and parameters: 1.85M. The computational complexity of our model is FLOPs: 97.23M and parameters: 1.92M. Models with lower complexity are less prone to overfitting when handling low fractional rate images, such as CIFAR-10 (32×32), compared with models with higher complexity. Meanwhile, in processing higher resolution images, such as CelebA (64×64), the less complex WGAN-GP (IS: 2.48 ± 0.03 and FID:11.07) does not perform as well as the more complex GIU-GANs (IS: 2.61 ± 0.02 and FID:6.34).

6. Conclusion

In this study, we proposed a new generative model called GIU-GANs, which incorporates a GIU module into the GANs framework. GIU-GANs place the GIU module comprising SE module and involution so that the model can exploit channel information to extract features and reduce redundant information to enhance the generated image quality. In addition, because RBN pays attention to the representative feature, we replace the BN of the generator near the input and output layers with RBN. Comparing GIU-GANs with other GANs, the IS and FID of GIU-GANs reached the state-of-the-art level on CIFAR-10 and CelebA datasets.

Considering the success of the Vision Transformer (VIT) (Dosovitskiy et al., 2021) model in visual tasks, we plan to combine the VIT model with GANs in the future to further improve performance. Moreover, involution does not change the number of output channels for the GIU-GANs to abandon convolution. How to improve involution so that GANs can neglect convolution and build a backbone only with involution is challenging and is considered future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research work is supported in part by the Fundamental Research Funds for the Central Universities under Grant 21621-017, in part by the Innovative Youth Program of Guangdong University, China under Grant 2019KQNCX194, and in part by the Educational and Scientifical Project of Guangdong Province, China 2021GXJK-
368.

References

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of machine learning research*: *vol. 70*, *International conference on machine learning* (pp. 214–223).

Bulat, A., Yang, J., & Tzimiropoulos, G. (2018). To learn image super-resolution, use a GAN to learn how to do image degradation first. In *Lecture notes in computer science*: *vol. 11210*, *European conference on computer vision* (pp. 187–202).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International conference on learning representations*.

Duchi, J. C., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Gao, S., Han, Q., Li, D., Cheng, M., & Peng, P. (2021). Representative batch normalization with feature calibration. In *IEEE conference on computer vision and pattern recognition* (pp. 8669–8679).

Ghosh, A., Kulharia, V., Namboodiri, V. P., Torr, P. H. S., & Dokania, P. K. (2018). Multi-agent diverse generative adversarial networks. In *2018 IEEE conference on computer vision and pattern recognition* (pp. 8513–8521).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial networks. CoRR abs/1406.2661.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein GANs. In *Advances in neural information processing systems* (pp. 5767–5777).

Hong, S., Yang, D., Choi, J., & Lee, H. (2018). Inferring semantic layout for hierarchical text-to-image synthesis. In *IEEE conference on computer vision and pattern recognition* (pp. 7986–7994).

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7132–7141).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *JMLR workshop and conference proceedings*: *vol. 37*, *International conference on machine learning* (pp. 448–456).

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. In *British machine vision conference*.

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *IEEE conference on computer vision and pattern recognition* (pp. 4401–4410).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), *3rd International conference on learning representations*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In Y. Bengio, & Y. LeCun (Eds.), *2nd International conference on learning representations*.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In M. Balcan, & K. Q. Weinberger (Eds.), *JMLR workshop and conference proceedings*, *Proceedings of the 33nd international conference on machine learning* (pp. 1558–1566).

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE conference on computer vision and pattern recognition* (pp. 105–114).

Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., et al. (2021). Involution: Inverting the inherence of convolution for visual recognition. CoRR abs/2103.06255.

Lin, M., Chen, Q., & Yan, S. (2014). Network in network. In *International conference on learning representations*.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *IEEE international conference on computer vision* (pp. 3730–3738).

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least squares generative adversarial networks. In *IEEE international conference on computer vision* (pp. 2813–2821).

Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. In *5th International conference on learning representations*.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. CoRR abs/1411.1784.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International conference on learning representations*.

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture notes in computer science*: *vol. 9912*, *European conference on computer vision* (pp. 483–499).

Nie, W., Narodytska, N., & Patel, A. (2019). RelGAN: Relational generative adversarial networks for text generation. In *International conference on learning representations*.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *International conference on learning representations*.

Reed, S. E., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. In *JMLR workshop and conference proceedings*: *vol. 48*, *International conference on machine learning* (pp. 1060–1069).

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. In *Advances in neural information processing systems* (pp. 2226–2234).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 2818–2826).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., et al. (2018). ESRGAN: Enhanced super-resolution generative adversarial networks. In *Lecture notes in computer science*: *vol. 11133*, *European conference on computer vision (ECCV) workshops* (pp. 63–79).

Zhang, H., Goodfellow, I. J., Metaxas, D. N., & Odena, A. (2019). Self-attention generative adversarial networks. In *Proceedings of machine learning research*: *vol. 97*, *International conference on machine learning* (pp. 7354–7363).

Zhang, H., Xu, T., & Li, H. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE international conference on computer vision* (pp. 5908–5916).

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., et al. (2019). StackGAN++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(8), 1947–1962.

Zhao, J. J., Mathieu, M., & LeCun, Y. (2017). Energy-based generative adversarial networks. In *5th International conference on learning representations*.

Zhu, J., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE international conference on computer vision* (pp. 2242–2251).