https://doi.org/10.26599/AIR.2022.9150005 CAAI Artificial Intelligence Research

Self-Sparse Generative Adversarial Networks

Wenliang Qian[1,2], Yang Xu[1,2], Wangmeng Zuo[3], and Hui Li[1,2]

ABSTRACT

Generative adversarial networks (GANs) are an unsupervised generative model that learns data distribution through adversarial training. However, recent experiments indicated that GANs are difficult to train due to the requirement of optimization in the high dimensional parameter space and the zero gradient problem. In this work, we propose a self-sparse generative adversarial network (Self-Sparse GAN) that reduces the parameter space and alleviates the zero gradient problem. In the Self-Sparse GAN, we design a self-adaptive sparse transform module (SASTM) comprising the sparsity decomposition and feature-map recombination, which can be applied on multi-channel feature maps to obtain sparse feature maps. The key idea of Self-Sparse GAN is to add the SASTM following every deconvolution layer in the generator, which can adaptively reduce the parameter space by utilizing the sparsity in multi-channel feature maps. We theoretically prove that the SASTM can not only reduce the search space of the convolution kernel weight of the generator but also alleviate the zero gradient problem by maintaining meaningful features in the batch normalization layer and driving the weight of deconvolution layers away from being negative. The experimental results show that our method achieves the best Fréchet inception distance (FID) scores for image generation compared with Wasserstein GAN with gradient penalty (WGAN-GP) on MNIST, Fashion-MNIST, CIFAR-10, STL-10, mini-ImageNet, CELEBA-HQ, and LSUN bedrooms datasets, and the relative decrease of FID is 4.76%–21.84%. Meanwhile, an architectural sketch dataset (Sketch) is also used to validate the superiority of the proposed method.

KEYWORDS

generative adversarial networks; self-adaptive sparse transform module; self-sparse generative adversarial network (Self-Sparse GAN)

Generative adversarial networks (GANs)[1] are a kind of unsupervised generation model based on game theory, and widely used to learn complex real-world distributions based on deep convolutional layers[2] (e.g. image generation). However, despite its success, training GANs is very unstable, and it may have problems such as gradient disappearance, divergence, and mode collapse[3,4]. The main reason is that training GANs needs to find a Nash equilibrium for a non-convex problem in a high dimensional continuous space[5]. In addition, it is pointed out that the loss function used in the original GANs[1] causes the zero gradient problem when there is no overlap between the generated data distribution and the real data distribution[6].

The stabilization of GAN training has been investigated by either modifying the network architecture[2,6–8] or adopting an alternative objective function[9–12]. However, these methods do not reduce the high-dimensional parameter space of the generator. When the task is complex (including more texture details and with high resolution), we often increase the number of convolution kernels to enhance the capability of the generator. Nevertheless, we do not exactly know how many convolution kernels are appropriate, which further increases the parameter space of the generator. Therefore, it is reasonable to speculate that a parameter redundancy exists in the generator. If the parameter space of the generator can be reduced, both the performance and training stability of GANs will be further improved.

Motivated by the aforementioned challenges and the sparsity in deep convolution networks[13,14], we propose a self-sparse generative adversarial network (Self-Sparse GAN), with a self-adaptive sparse transform module (SASTM) after each deconvolution layer. The SASTM consisting of the sparsity decomposition and feature-map recombination is applied on multi-channel feature maps of the deconvolution layer to obtain sparse feature maps. The channel sparsity coefficients and position sparsity coefficients are obtained by using a two-headed neural network to transform the latent vector in the sparsity decomposition. Then, the sparse multi-channel feature maps are acquired by a superposition of the channel sparsity and position sparsity, which can be obtained by the feature maps multiplying the corresponding sparsity coefficients. The corresponding sparsity coefficients will alleviate the zero gradient problem by maintaining meaningful features in the batch normalization (BN) layer and driving the weights of deconvolution layers away from being negative. Meanwhile, the sparse feature maps will free some of the convolution kernels, that is, the weights do not affect the model, thus reducing the parameter space.

Our contributions. We propose a novel Self-Sparse GAN, in which the training of generator considers the adaptive sparsity in multi-channel feature maps. We use the SASTM to adaptively implement the sparsity of the feature map, and theoretically prove that our method not only reduces the search space of the convolution kernel weight but also alleviates the zero gradient problem. We evaluate the performance of proposed Self-Sparse GAN using the MNIST[15], Fashion-MNIST[16], CIFAR-10[17], STL-10[18], mini-ImageNet[19], CELEBA-HQ[7], LSUN bedrooms[20], and

1 Labortoray of Artificial Intelligence, Harbin Institute of Technology, Harbin 150001, China
2 School of Civil Engineering, Harbin Institute of Technology, Harbin 150090, China
3 School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
Address correspondence to Hui Li, lihui@hit.edu.cn

Sketch datasets. The experimental results show that our method achieves the best Fréchet inception distance (FID) scores for image generation compared with Wasserstein GAN with gradient penalty (WGAN-GP)[11], and the relative decrease of FID[21] is 4.76%–21.84%.

1 Related Work

Generative adversarial network. GANs[1] can learn the data distribution through the game between the generator and discriminator, and have been widely used in image generation[22], video generation[23], image translation[24], and image inpainting[25].

Optimization and training frameworks. With the development of GANs, more and more researchers are committed to settling the training barriers of gradient disappearance, divergence, and mode collapse. In Ref. [5], noise is added to the generated data and real data to increase the support of two distributions and alleviate the problem of gradient disappearance. In Ref. [10], the least squares loss function is adopted to stabilize the training of the discriminator. Wasserstein GAN (WGAN)[9] uses the earth mover's distance (EMD) instead of the Jensen-Shannon divergence in the original GAN, which requires the discriminator to satisfy the Lipschitz constraint and can be achieved by weight clipping. Because weight clipping will push weights towards the extremes of the clipping range, WGAN-GP[11] uses the gradient penalty to make the discriminator satisfy the Lipschitz constraint. Another way to enforce the Lipschitz constraint is proposed in Ref. [12] by spectral normalization. A series of adaptive methods with the transformation of the latent vector to get additional information are also widely used in GANs. Reference [26] uses an adaptive affine transformation to utilize spatial feedback from the discriminator to improve the performance of GANs. Reference [27] uses a nonlinear network to transform the latent space to obtain the intermediate latent space, which controls the generator through adaptive instance normalization (AdaIN) in each convolutional layer. In Ref. [28], a nonlinear network is used to transform the latent vector to obtain the affine transformation parameters of the BN layer to stabilize the GAN training. Sparse generative adversarial network (SPGAN)[29] creates a sparse representation vector for each image patch and then synthesizes the entire image by multiplying generated sparse representations to a pre-trained dictionary and assembling the resulting patches. Based on a GAN, Liu et al.[30] proposed Task-Oriented GAN to tackle difficulties in PolSAR image interpretation, including PolSAR data analysis and small sample problems.

Sparsity in convolutional neural networks. Deep convolution networks have made great progress in a wide range of fields, especially for image classification[31]. However, there is a strong correlation between the performance of the network and the network size[32], which also leads to parameter redundancy in deep convolutional networks. The sparse convolutional neural networks[13] uses sparse decomposition of the convolution kernels to reduce more than 90% parameters, while the drop of accuracy is less than 1% on the ILSVRC2012 dataset. Reference [14] proposes L_0 norm regularization for neural networks to encourage weights to become exactly zero to speed up training and improve generalization.

A new feature selection algorithm is proposed, where the algorithm utilizes the residual term in sparse regression to ensure that the learned low-dimensional subspaces have greater fault tolerance[33]. Shang et al.[34] proposed a new feature selection method (named SLMEA), which combines the sparse transform representation with pseudo-label matrix learning to guide the learning of sparse low-dimensional space.

2 Self-Sparse GAN

Motivated by the aforementioned challenges, we aim to design the generator with a mechanism, which can use fewer feature maps to learn useful representations. Inspired by the dual attention network (DANet)[35], we first design a two-headed neural network to transform the latent vector to obtain the channel sparsity coefficient and position sparsity coefficient of the multi-channel feature maps. Second, we multiply the multi-channel feature maps by the channel sparse coefficient and position sparse coefficient, respectively. Then, we add the results to get the output of SASTM.

The differences and improvements between the SASTM and DANet are as follows: (1) SASTM is used for the image generation task, while DANet is used for the scene segmentation task; (2) SASTM can obtain sparse multi-channel feature maps by the proposed sparsity decomposition and feature-map recombination; and (3) SASTM can adaptively reduce the parameter space and alleviate the zero gradient problem.

The proposed Self-Sparse GAN adds an SASTM behind each deconvolution layer of the generator. Self-Sparse GAN only modifies the architecture of the generator, and its conceptual diagram is shown in Fig. 1.

We define the process of transforming the planar size of the feature map from $H\times W$ to $2H\times 2W$ as a generative stage. For example, when the pixel resolution of the generated image is 128×128, the hierarchical processes of feature map generation $z\rightarrow 4\times 4\rightarrow 8\times 8\rightarrow 16\times 16\rightarrow 32\times 32\rightarrow 64\times 64\rightarrow 128\times 128$ are defined as different stages in the generator. Stage $t=3$ refers to $8\times 8\rightarrow 16\times 16$, where $t\in\{1, 2, 3, 4, 5, 6\}$ and $T=6$ denotes the total number of stages.

2.1 SASTM: Self-adaptive sparse transform module

SASTM includes the sparsity decomposition and feature-map recombination. The sparsity decomposition consists of channel sparsity module (CSM) and position sparsity module (PSM) to obtain the channel sparsity coefficient and position sparsity coefficient.

As illustrated in Fig. 2, a two-headed neural network will be employed to obtain the corresponding sparsity coefficients. In the two-headed neural network, the underlying shared layers multilayer perceptron (MLP) are defined as f^t, and the exclusive networks are g_1^t and g_2^t, respectively. $\alpha_i^t\geqslant 0$ and $\beta_{j,k}^t\geqslant 0$ can be obtained as follows:

$$\alpha^t = ReLU(g_1^t(f^t(z))) \tag{1}$$

$$\beta^t = ReLU(g_2^t(f^t(z))) \tag{2}$$

where Eqs. (1) and (2) represent CSM and PSM, respectively.

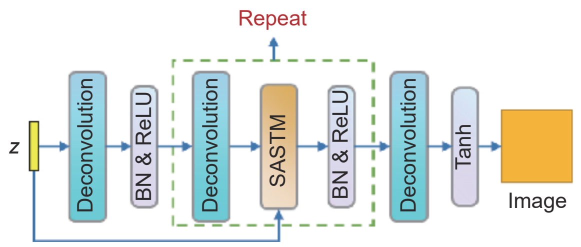


Fig. 1 Concept diagram of the modified generator in the Self-Sparse GAN. Deconvolution, BN, ReLU, and Tanh represent the deconvolutional layer, the batch normalization layer, rectified linear activation function, and tanh activation function, respectively.

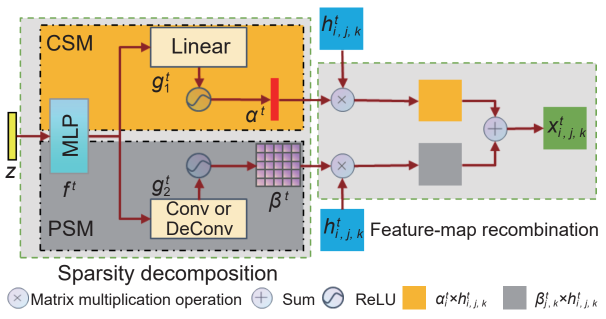


Fig. 2 Concept diagram of the self-adaptive sparse transform module.

$\alpha_i^t \in \mathbb{R}^{C^t}$ and $\beta_{j,k}^t \in \mathbb{R}^{H^t \times W^t}$ are coefficients of the channel sparsity and position sparsity, respectively. When $\alpha_i^t = 0$ and $\beta_{j,k}^t = 0$, the corresponding channel and spatial location will become useless, respectively.

Suppose that the output after the deconvolution layer is $h_{i,j,k}^t \in \mathbb{R}^{C^t \times H^t \times W^t}$, where C^t, H^t, and W^t represent the number of channels, height, and width of the feature maps, respectively. The feature-map recombination will be calculated as follows:

$$x_{i,j,k}^t = \alpha_i^t \times h_{i,j,k}^t + \beta_{j,k}^t \times h_{i,j,k}^t \tag{3}$$

where $x_{i,j,k}^t \in \mathbb{R}^{C^t \times H^t \times W^t}$ is sparse feature maps. Therefore, SASTM is the superposition of channel sparsity and position sparsity.

The sparse rate of position sparsity in the i-th channel is defined as follows:

$$\xi_i^t = \frac{crad\left(\left\{x_{i,j,k}^t \middle| x_{i,j,k}^t = 0\right\}\right)}{H^t W^t} \tag{4}$$

where $crad\ (\cdot)$ is used to signify number of elements in the set. If $\xi_i^t \geqslant 2/3$, the i-th channel is regarded to be sparse.

The sparse rate of the channel sparsity is defined as follows:

$$\zeta = \frac{crad\left(\left\{\xi_i^t \middle| \xi_i^t \geqslant 2/3\right\}\right)}{C^t} \tag{5}$$

In the back propagation (BP) process, the derivatives of loss function with respect to $h_{i,j,k}^t$, α_i^t, and $\beta_{j,k}^t$ are calculated as follows:

$$\nabla_{h_{i,j,k}^t} = \nabla_{x_{i,j,k}^t}\left(\frac{\partial x_{i,j,k}^t}{\partial h_{i,j,k}^t}\right) = \nabla_{x_{i,j,k}^t}(\alpha_i^t + \beta_{j,k}^t),$$

$$\nabla_{\alpha_i^t} = \nabla_{x_{i,j,k}^t}\left(\frac{\partial x_{i,j,k}^t}{\partial \alpha_i^t}\right) = \nabla_{x_{i,j,k}^t} h_{i,j,k}^t,$$

$$\nabla_{\beta_{j,k}^t} = \nabla_{x_{i,j,k}^t}\left(\frac{\partial x_{i,j,k}^t}{\partial \beta_{j,k}^t}\right) = \nabla_{x_{i,j,k}^t} h_{i,j,k}^t \tag{6}$$

2.2 Mechanism analysis of SASTM

To analyze the role of the SASTM, we select the t-th generation stage as shown in Fig. 3.

For the convenience of discussion, we assume that the dimensions of the input and output feature maps of the deconvolution layer remain the same ($C^t \times H^t \times W^t$).

Meanwhile, the size of the deconvolution kernel is 1×1. The feedforward process is expressed as follows:

$$x_{i,j,k}^t = \left(\alpha_i^t \sum_m v_{m,i}^t y_{m,j,k}^{t-1} + \beta_{j,k}^t \sum_m v_{m,i}^t y_{m,j,k}^{t-1}\right),$$

$$y_{i,j,k}^t = \max\left\{\varphi_{\mathrm{bn}}^t(x_{i,j,k}^t), 0\right\} \tag{7}$$

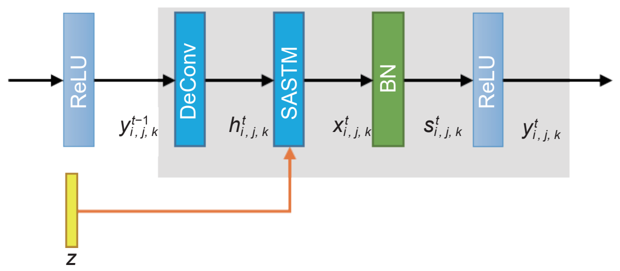


Fig. 3 The concept diagram of the t-th generation stage. $y_{i,j,k}^{t-1}$, $h_{i,j,k}^t$, $x_{i,j,k}^t$, $s_{i,j,k}^t$ and $y_{i,j,k}^t$ represent the output feature maps of the previous ReLU, deconvolution, SASTM, BN, and ReLU layers, respectively, and their dimensions are $C^t \times H^t \times W^t$.

where $v_{m,i}^t$ denotes the corresponding deconvolution weight, $y_{m,j,k}^{t-1}$ denotes the element of position j,k in the m-th channel, and φ_{bn}^t denotes the BN operation.

In the BP process, the derivatives of the loss function with respect to $v_{m,i}^t$, α_i^t, and $\beta_{j,k}^t$ are calculated as follows:

$$\nabla_{v_{m,i}^t} = \begin{cases} \nabla_{y_{i,j,k}^t}(\alpha_i^t + \beta_{j,k}^t) y_{m,j,k}^{t-1}, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) > 0; \\ 0, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) \leqslant 0 \end{cases} \tag{8}$$

$$\nabla_{\alpha_i^t} = \begin{cases} \nabla_{y_{i,j,k}^t} \sum_m v_{m,i}^t y_{m,j,k}^{t-1}, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) > 0; \\ 0, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) \leqslant 0 \end{cases} \tag{9}$$

$$\nabla_{\beta_{j,k}^t} = \begin{cases} \nabla_{y_{i,j,k}^t} \sum_m v_{m,i}^t y_{m,j,k}^{t-1}, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) > 0; \\ 0, & \varphi_{\mathrm{bn}}^t\left(x_{i,j,k}^t\right) \leqslant 0 \end{cases} \tag{10}$$

Intuitively, whether α_i^t and $\beta_{j,k}^t$ are equal to zero or greater than zero will remain unchanged after certain training steps. Therefore, we make the following assumption.

Hypothesis 1. When α_i^t and $\beta_{j,k}^t$ have represented the significance of channel and position sparsity, their signs will remain unchanged.

In this section, we prove that the proposed SASTM plays the following three roles.

(1) Reducing the search space of convolution parameters in the generator. In Eq. (7), the dimension of the deconvolution kernel weight $v_{m,i}^t$ is $C^t \times C^t$. Find $\forall i \in A \subset \{1,2,\ldots,C^t\}$, where $\alpha_i^t = 0$, and $\forall j \in \{1,2,\ldots,H^t\}$, $\forall k \in \{1,2,\ldots W^t\}$, where $\beta_{j,k}^t = 0$ then $x_{i_1,j,k}^t = 0, \nabla_{v_{m,i_1}^t} = 0$ for all times from Hypothesis 1, which indicates that the i-th channel will no longer work in both feedforward and backward processes in training. Consequently, the dimensions of the valid convolutional kernels are $(C^t - |A|) \times C^t$, thus reducing the search space of the convolutional parameters in the generator.

(2) Maintaining meaningful features in the BN layer to alleviate the zero gradient problem. For the convenience of discussion, we do not consider the affine transformation in the BN layer. At the same time, because dividing by the standard deviation in the BN layer will not change the sign, we also ignore the standard deviation in the BN layer for the remainder of the discussion. $x_{i,j,k}^t$ can be divided into two parts $\{x_{i,j,k}^t | x_{i,j,k}^t < 0\}$ and $\{x_{i,j,k}^t | x_{i,j,k}^t \geqslant 0\}$. When $\{x_{i,j,k}^t | x_{i,j,k}^t < 0\}$ passes through the BN layer, a part of its value will become greater than zero in feedforward and thus mitigate the zero gradient problem in backward. Here, we ignore the part still less than zero. Therefore, in the following discussion, we assume $x_{i,j,k}^t \geqslant 0$, and denote the positive values as $x_{i,d,e}^t = \{\tilde{x}_{i,d,e}^t | x_{i,d,e}^t > 0\}$, $d \in \{1,2,\ldots,D\}$, $e \in \{1,2,\ldots,E\}$.

According to the definition of the sparse rate of position sparsity, $DE = \left(1-\xi_i^t\right) H^t W^t$. Therefore, the computation of the BN layer can be expressed as

$$s_{i,j,k}^t = x_{i,j,k}^t - \frac{1}{H^t W^t}\sum x_{i,j,k}^t = x_{i,j,k}^t - (1-\xi_i^t)\frac{1}{DE}\sum \tilde{x}_{i,d,e}^t = x_{i,j,k}^t - (1-\xi_i^t)\tilde{\mu}_i^t \quad (11)$$

where $\tilde{\mu}_i^t = \sum \tilde{x}_{i,d,e}^t / DE > 0$ and $s_{i,j,k}^t$ is the value of $x_{i,j,k}^t$ after passing through BN.

The conditional probability of $\left(s_{i,j,k}^t < 0 | x_{i,j,k}^t > 0\right)$ is $P\left(\tilde{x}_{i,d,e}^t - \tilde{\mu}_i^t < 0\right)$. In addition, considering the position sparsity in $x_{i,j,k}^t$, the aforementioned probability is approximated as $P\left(\tilde{x}_{i,d,e}^t - (1-\xi_i^t)\tilde{\mu}_i^t < 0\right)$, where

$$P\left(\tilde{x}_{i,d,e}^t - \tilde{\mu}_i^t < 0\right) = P\left(\tilde{x}_{i,d,e}^t - (1-\xi_i^t)\tilde{\mu}_i^t < \xi_i \tilde{\mu}_i^t\right) > P\left(\tilde{x}_{i,d,e}^t - (1-\xi_i^t)\tilde{\mu}_i^t < 0\right) \quad (12)$$

From Eq. (12), when position sparsity exits in $x_{i,j,k}^t$, the probability of $s_{i,j,k}^t < 0$ will decrease, which increases the probability $y_{i,j,k}^t > 0$ from Eq. (7). In addition, a larger ξ_i^t will lead to a lower probability of $s_{i,j,k}^t < 0$. Therefore, the gradient in the backpropagation will not disappear. In other words, when α_i^t and $\beta_{j,k}^t$ have already determined the sparse channels and spatial locations, SASTM will reduce the likelihood that useful feature information is dropped after passing through the BN layer, thus maintaining meaningful features to alleviate the zero gradient problem.

(3) Driving the convolutional weights away from being negative. From the above derivation, the probability that $x_{i,d,e}^t$ is less than zero after passing through the BN layer can be reduced when the proposed SASTM is implemented into the network, and a larger position sparsity rate ξ_i^t leads to a smaller probability. Therefore, for convenience, we will not consider the BN layer in the discussion below.

When $\nabla_{y_{i,j,k}^t} > 0$ and $x_{i,j,k}^t > 0$, it can be inferred $\nabla_{v_{m,i}^t} > 0$ from Eq. (8), and then $v_{m,i}^t$ will decrease in the gradient descent algorithm. Similarly, according to Eqs. (9) and (10), $\nabla_{\alpha_i^t} > 0$ and $\nabla_{\beta_{j,k}^t} > 0$ can be inferred, and thus α_i^t and $\beta_{j,k}^t$ will decrease. However, from Hypothesis 1, α_i^t and $\beta_{j,k}^t$ will not be less than zero. According to Eq. (8), $\nabla_{v_{m,i}^t}$ is obtained by the factors α_i^t and $\beta_{j,k}^t$. Therefore, the decrease of $v_{m,i}^t$ is suppressed.

When $\nabla_{y_{i,j,k}^t} < 0$ and $x_{i,j,k}^t > 0$, it can be inferred $\nabla_{v_{m,i}^t} < 0$ from Eq. (8), and then $v_{m,i}^t$ will increase. Similarly, according to Eq. (9) and Eq. (10), $\nabla_{\alpha_i^t} < 0$ and $\nabla_{\beta_{j,k}^t} < 0$ can be inferred, and thus α_i^t and $\beta_{j,k}^t$ will increase. Therefore, the increase of $v_{m,i}^t$ is promoted.

Therefore, the proposed SASTM enables to drive convolutional weights away from being negative. This phenomenon has been similarly reported as the channel scaling layer[36].

3 Experiment and Discussion

3.1 Baseline model: WGAN-GP

WGAN-GP[11] has good theoretical and stability properties in practice, and a zero-centered gradient penalty further enhances the convergency[3]. Therefore, WGAN with a zero-centered gradient penalty is adopted as the baseline for comparison with our method, and the objective function is as follows.

$$\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim P_r}\left[D(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{z}\sim P_z}\left[D(G(\boldsymbol{z}))\right] - \lambda \cdot \mathbb{E}_{\widehat{\boldsymbol{x}}\sim P_{\widehat{x}}}\left[\left\|\nabla_{\widehat{x}} D(\widehat{\boldsymbol{x}})\right\|_2^2\right] \quad (13)$$

where $\mathbb{E}$ denotes the mean symbol, and $\boldsymbol{x}$, $\boldsymbol{z}$, λ and $\widehat{\boldsymbol{x}}$ represent the real data, latent vector, gradient penalty coefficient, and random samples with sampling uniformly along straight lines between pairs of real data and fake data[11].

3.2 Experimental settings

We test the proposed Self-Sparse GAN using a DCGAN-like[2] network architecture on the following datasets: (1) MNIST: 60 thousand grayscale images; (2) Fashion-MNIST: 60 thousand grayscale images; (3) CIFAR-10: 60 thousand RGB images; (4) STL-10: 100 thousand RGB images; (5) mini-ImageNet: 60 thousand RGB images; (6) CELEBA-HQ: 30 thousand RGB images; and (7) LSUN bedrooms: 3 million RGB images. Details of Self-Sparse GAN are referred to the Appendix. The investigated resolutions of the generated images are listed in Table 1.

We use the Adam[37] optimizer and set the learning rates of generator and discriminator as 0.0001 and 0.0003 on all datasets as suggested in Ref. [21]. Because multiple discriminator steps per generator step can help the GAN training in WGAN-GP, we set two discriminator steps per generator step for 100 thousand generator steps. We set betas (0.5, 0.999) for MNIST, Fashion-MNIST, CIFAR-10, STL-10, and mini-ImageNet, and (0.0, 0.9) for CELEBA-HQ and LSUN bedrooms.

For the evaluation of model sampling quality, we use FID[21] as the evaluation metric, which can measure the distance between the real and generated data distributions. A smaller FID indicates better qualities of the generated images. The FID is calculated as

$$FID(\boldsymbol{x},\boldsymbol{g}) = \left\|\mu_{\boldsymbol{x}} - \mu_{\boldsymbol{g}}\right\|_2^2 + Tr\left(var(\boldsymbol{x}) + var(\boldsymbol{g}) - 2\sqrt{var(\boldsymbol{x})var(\boldsymbol{g})}\right) \quad (14)$$

where $Tr\,(\cdot)$, μ, and $var\,(\cdot)$ denote the trace of a matrix, mean, and convariance, and $\boldsymbol{x}$ and $\boldsymbol{g}$ denote the real and generated data, respectively. To obtain training curves quickly, the FID is evaluated every 500 generator steps using the 5 thousand samples.

3.3 Results

Figure 4 shows two representative FID curves on MNIST and CIFAR-10. Figure 4 indicates that our method converges faster in the same FID level. Table 2 shows the mean and standard deviation of the best FIDs on all datasets. Experimental results show that our method reduces FIDs on all datasets and the relative decrease of FID is 4.76%–21.84%. Although the Self-Sparse GAN does not significantly exceed the baseline on CELEBA-HQ with the resolutions of 64×64×3, the relative decrease of FID is still close to 5%. Meanwhile, these results demonstrate that our method can both improve the generation quality of grayscale and RGB images. In addition, the relative improvement of model performance increases with the resolution of generated images from $64\times64\times3$ to $128\times128\times3$ on CELEBA-HQ and LSUN bedrooms.

Table 1 Investigated pixel resolutions of generated images.

Dataset	Pixel resolution of the generated image
MNIST	32×32, 128×128
Fashion-MNIST	64×64, 128×128
CIFAR-10	128×128×3
STL-10	64×64×3
mini-ImageNet	32×32×3, 64×64×3
CELEBA-HQ	64×64×3, 128×128×3
LSUN bedrooms	64×64×3, 128×128×3

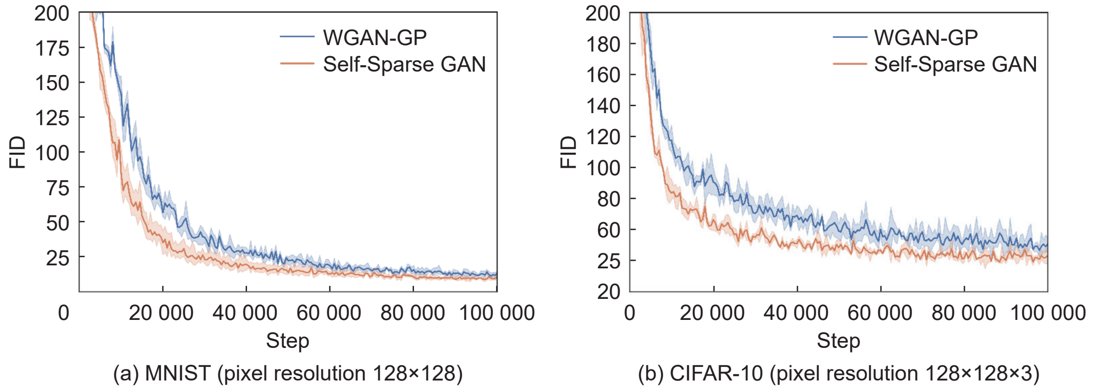


Fig. 4 FID training curves on MNIST and CIFAR-10, depicting the mean performance of three random trainings with a 95% confidence interval.

Table 2 Comparison of FIDs between our proposed Self-Sparse GAN and the baseline WGAN-GP. The mean and standard deviation of the FID are calculated through three individual trainings with different random seeds.

Dataset	Pixel resolution	Model	FID	Dataset	Pixel resolution	Model	FID
MNIST (Grayscale)	32×32×1	WGAN-GP	7.43 ± 0.28	mini- ImageNet (RGB)	32×32×3	WGAN-GP	33.16 ± 0.02
		Self-Sparse GAN	6.26 ± 0.64			Self-Sparse GAN	28.88 ± 0.37
	128×128×1	WGAN-GP	10.42 ± 0.86		64×64×3	WGAN-GP	58.81 ± 3.28
		Self-Sparse GAN	8.32 ± 1.03			Self-Sparse GAN	54.78 ± 0.28
Fashion- MNIST (Grayscale)	64×64×1	WGAN-GP	20.37 ± 0.87	CELEBA- HQ (RGB)	64×64×3	WGAN-GP	15.95 ± 0.44
		Self-Sparse GAN	15.92 ± 1.10			Self-Sparse GAN	15.19 ± 0.18
	128×128×1	WGAN-GP	20.41 ± 0.70		128×128×3	WGAN-GP	32.40 ± 2.03
		Self-Sparse GAN	17.67 ± 0.87			Self-Sparse GAN	27.72 ± 1.54
CIFAR-10 (RGB)	128×128×3	WGAN-GP	43.77 ± 2.10	LSUN bedrooms (RGB)	64×64×3	WGAN-GP	59.12 ± 0.95
		Self-Sparse GAN	36.69 ± 1.53			Self-Sparse GAN	55.06 ± 2.00
STL-10 (RGB)	64×64×3	WGAN-GP	63.88 ± 1.33		128×128×3	WGAN-GP	102.16 ± 0.85
		Self-Sparse GAN	56.23 ± 1.38			Self-Sparse GAN	84.78 ± 2.89

3.4 Ablations

To investigate the effects of CSM and PSM in the proposed SASTM, we perform ablation studies on Fashion-MNIST and STL-10. “Without PSM” represents using CSM only and $\beta_{j,k}^{t}=0$. Similarly, “without CSM” represents using PSM only and $\alpha_{i}^{t}=0$.

As shown in Table 3, the model performance has a significant improvement on Fashion-MNIST and STL-10 when both CSM and PSM are applied. Since the position sparsity coefficient β^{t} is shared by all channels, it is difficult to represent the pixel-wise sparsity among different channels without α^{t}. Therefore, using only PSM may not function well. Furthermore, when only CSM is used, the model may lack generation power. Figure 5 also shows that using only CSM on the Fashion-MNIST dataset causes the multi-channel feature maps too sparse, which will suppress the model performance.

Table 3 Comparisons of FIDs in ablation studies on Fashion-MNIST and STL-10.

Dataset	Pixel resolution	Method	FID
Fashion-MNIST	128×128×1	WGAN-GP	20.41 ± 0.70
		Self-Sparse GAN	17.67 ± 0.87
		Without PSM	21.51 ± 0.29
		Without CSM	163.71±3.99
STL-10	64×64×3	WGAN-GP	63.88 ± 1.33
		Self-Sparse GAN	56.23 ± 1.38
		Without PSM	60.85 ± 0.99
		Without CSM	64.87 ± 1.71

Robustness to hyperparameters of Adam training. GANs are very sensitive to hyperparameters of the optimizer. Therefore, we evaluate different hyperparameter settings to validate the robustness of our method. We test two popular settings of betas in Adam: (0, 0.9) and (0.5, 0.999). Table 4 compares the mean and standard deviation of FID scores on STL-10. It suggests that the proposed Self-Sparse GAN consistently improves model performance.

Robustness to network architectures. To further test the robustness of the proposed Self-Sparse GAN to different network architectures, we use two common network architectures from DCGAN and ResNet on STL-10. Tables 5 and 6 give the details. Table 7 shows the FID scores using different network architectures on STL-10, which shows that our method is robust to both DCGAN and ResNet network architectures.

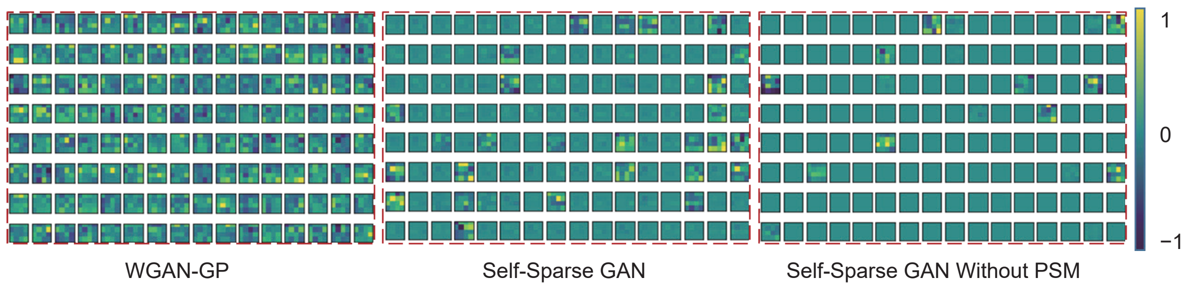


Fig. 5 Ablation study. Using only CSM causes too sparse multi-channel feature maps. The colorimetric scale represents the values of the pixel points in a feature map.

Table 4 Comparisons of FID in the robustness experiments on STL-10 with different Adam hyperparameter settings.

Dataset	Betas	Method	FID
STL-10	0, 0.9	WGAN-GP	63.88 ± 1.33
	0, 0.9	Self-Sparse GAN	**56.23 ± 1.38**
	0.5, 0.999	WGAN-GP	67.13 ± 0.77
	0.5, 0.999	Self-Sparse GAN	**56.51 ± 1.62**

Table 5 The SASTM Architecture on STL-10 with 128×128×3 resolution.

Layer	Operator
f	Linear: $z \in \mathbb{R}^{128} \sim N(0,1) \rightarrow 256$, ReLU
g_1	Linear: $256 \rightarrow M_c$, ReLU
g_2	Conv or DeConv. $16 \times 16 \rightarrow M_h \times M_w$, ReLU

Table 6 ResNet Generator and Discriminator in the robustness experiment on STL-10.

Model		Layer
Generator	First block	$z \in \mathbb{R}^{128} \sim N(0,1)$ DeConv. 4×4, stride =1, padding =0, **SASTM**
	ResNet block	BN-512, ReLU, Upsample-2, Conv. 3×3, stride=1, padding=1, BN-256, ReLU, Conv. 3×3, stride=1, padding=1, **SASTM**
	ResNet block	BN-256, ReLU, Upsample-2, Conv. 3×3, stride=1, padding=1, BN-128, ReLU, Conv. 3×3, stride=1, padding=1, **SASTM**
Discriminator		Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU, Conv. 3×3, stride=1, padding=1, IN-128, LeakyReLU, Conv. 3×3, stride=1, padding=1, Downsample-0.5, IN-128, LeakyReLU, Conv. 3×3, stride=1, padding=1, IN-256, LeakyReLU, Conv. 3×3, stride=1, padding=1, Downsample-0.5

3.5 Visualization of SASTM features

To illustrate the function of SASTM, we visualize the multi-channel feature maps of each deconvolution layer in the generator on MNIST with a pixel resolution of 128×128. Figure 6 shows multi-channel feature maps in all representative levels of 64×64, $32 \times 32, 16 \times 16, 8 \times 8$, and 4×4.

Table 7 Comparisons of FID in the robustness experiments on STL-10 with different network architectures.

Dataset	Architecture	Method	FID
STL-10	DCGAN	WGAN-GP	63.88 ± 1.33
		Self-Sparse GAN	**56.23 ± 1.38**
	ResNet	WGAN-GP	65.16 ± 5.96
		Self-Sparse GAN	**60.31 ± 4.29**

Results show that the proposed Self-Sparse GAN learns to pick useful sparse convolutional kernels instead of using all kernels greedily. It also proves that our method can obtain sparse multi-channel feature maps, thus reducing network parameters.

Validation of Hypothesis 1. We can verify this hypothesis by visualizing feature maps under different training steps on MNIST, as shown in Fig. 7. The results illustrate that the sign of α_i^t and $\beta_{j,k}^t$ will remain unchanged after 5 000 generator steps. This phenomenon verifies the Hypothesis 1.

3.6 Investigation of relationship between sparsity and FID

In Section 2.2, we have proved that the proposed SASTM will alleviate the zero gradient problem and thus improve the model performance. To analyze the relationship between model sparsity and FID quantitatively, we define the average position sparsity rate $\bar{\xi}$ of the generators as

$$\bar{\xi} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{C^t}\sum_{i=1}^{C^t}\xi_i^t\right) \tag{15}$$

We select the same network architecture to calculate the corresponding average position sparsity rate according to Eq. (15), as shown in Table 8. From the data, it indicates that a higher average position sparsity rate may lead to a greater improvement in FID except on the mini-ImageNet with 64×64 resolution and CIFAR-10 with 128×128 resolution, which will be further investigated in the future study. The Pearson's coefficient is used to measure the correlation between the average position sparsity rate and FID as

$$\rho(\bar{\xi}, \eta) = \frac{\mathbb{E}\left[\left(\bar{\xi} - \mu_{\bar{\xi}}\right)\left(\eta - \mu_{\eta}\right)\right]}{\sigma_{\bar{\xi}}\sigma_{\eta}} \tag{16}$$

where $\bar{\xi}$ and η denote the average position sparsity rate and the relative decrease of FID, respectively. A positive correlation between the average position sparsity rate and FID is found. When the pixel resolution of the generated image is 64×64, Pearson's correlation coefficient is 0.62. When the pixel resolution of the generated image is 128×128, Pearson's correlation coefficient is 0.79. Meanwhile, with the increase in resolution, Pearson's correlation coefficient will increase.

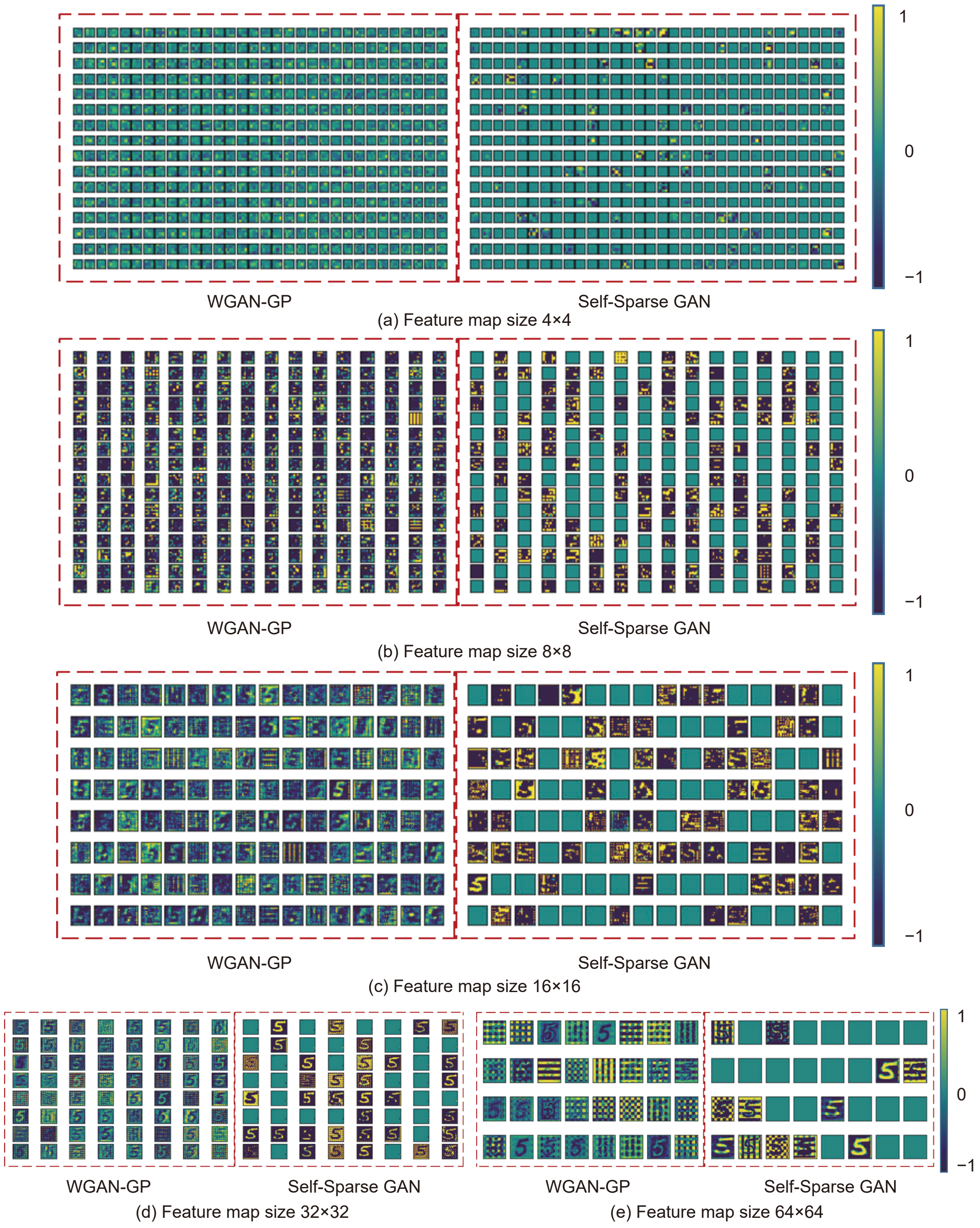


Fig. 6 Visualization of the feature map of the SASTM output. The colorimetric scale represents the values of the pixel points in a feature map. It can be observed that Self-Sparse GAN learns to pick useful convolutional kernels instead of using all convolutional kernels for image generation. In (a), we can observe that some sparse feature maps have regular feature points, which means that the PSM is working.

3.7 Sketch datasets

Architectural art shape sketches reflect the understanding of design goals, art styles, and local culture, which can not only present the designer's initial design ideas but also inspire new afflatus in turn. However, in contrast to other types of images, architectural shape sketches are composed of lines or edges, and contain significant amounts of blank space. Thus, sparsity is supposed to exist in a sketch image, which can be used to validate the superiority of the proposed Self-Sparse GAN. To this end, we collected building shape images and processed them to obtain a Sketch dataset of building shapes. The network architectures of Self-Sparse GAN and WGAN-GP are the same as Fashion-MNIST and the pixel resolution is 128×128.

Table 9 gives the FID scores of WGAN-GP and Self-Sparse GAN on the Sketch dataset, which shows that our method also

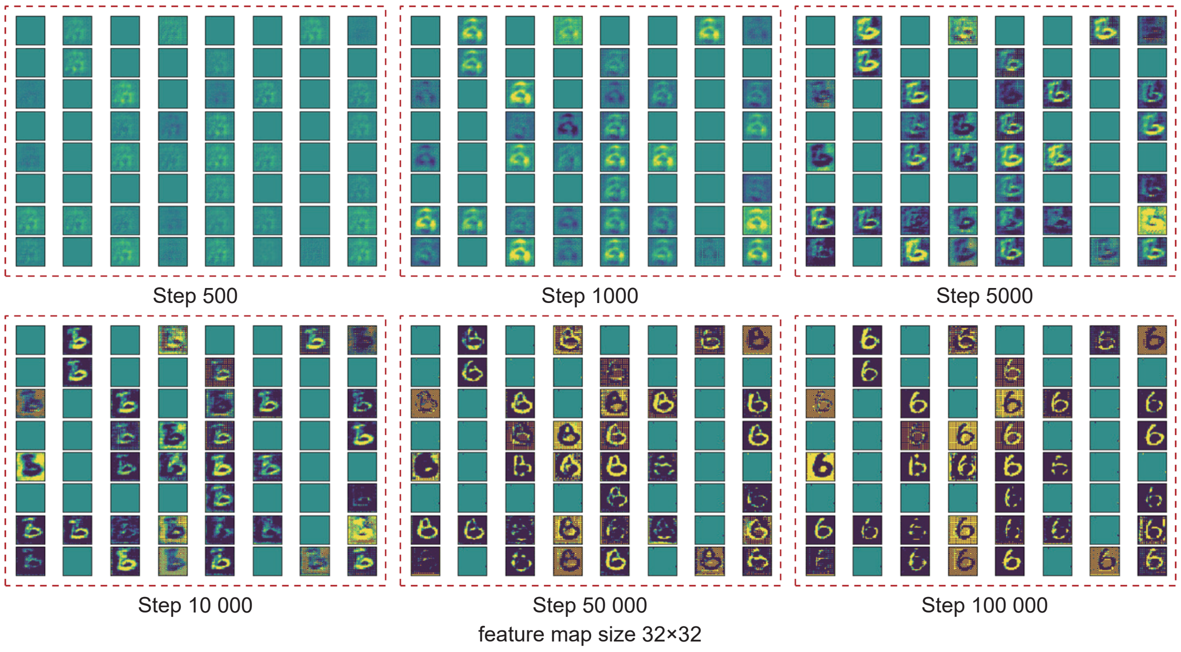


Fig. 7 Validation of Hypothesis 1 on MNIST with the pixel resolution of 128×128.

Table 8 Relationship between sparseness and FID with the same network architecture.

Pixel resolution	Dataset	Average position sparsity rate	FID reduction (%)
64×64	STL-10	0.44	11.97
	mini-ImageNet	0.47	6.83
	CELEBA-HQ	0.26	4.76
	LSUN bedrooms	0.39	6.86
128×128	MNIST	0.53	20.21
	CIFAR-10	0.47	16.17
	CELEBA-HQ	0.21	14.43
	LSUN bedrooms	0.31	17.02

Table 9 FIDs with WGAN-GP and Self-Sparse GAN on the Sketch.

Dataset	Method	FID
Sketch	WGAN-GP	76.68 ± 1.08
	Self-SparseGAN	67.58 ± 0.89

outperforms WGAN-GP on the architectural sketch dataset. Figure 8 gives the sketches generated by the proposed Self-Sparse GAN and WGAN-GP. The results show that these sketches generated by WGAN-GP often lack local details and realistic textures. For example, it can be observed that meaningful architectural features are not found in the first sketch by WGAN-GP in Fig. 8a, and the fifth sketch by WGAN-GP in Fig. 8a learns the outline of the building, but the local details and realistic textures are missing. In addition, blurring often occurs in the generated sketches with a much greater probability when using the WGAN-GP as the generator. These results also illustrate that the proposed method can perform the intelligent design of architectural sketches. More details of the autonomous early-stage design of architectural sketches by using the Self-Sparse GAN can be found in Ref. [38].

To further illustrate the superiority of our method, we selected the self-attention generative adversarial networks (SAGAN)[8] for comparison in CELEBA-HQ with the pixel resolution of 128×128×3 and Sketch[38] with the pixel resolution of 128×128×1. Table 10 shows the comparison of WGAN-GP, SAGAN, and Self-Sparse GAN, indicating our method is better than both WGAN-

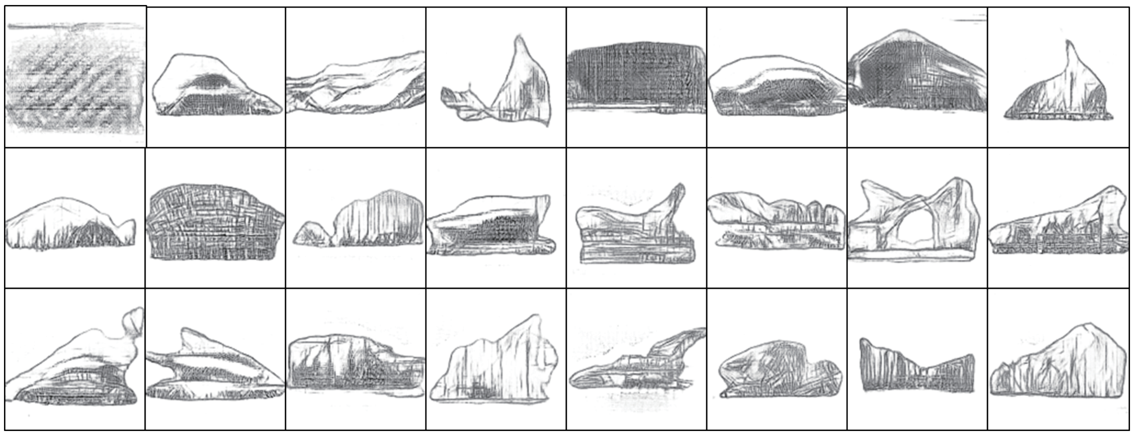
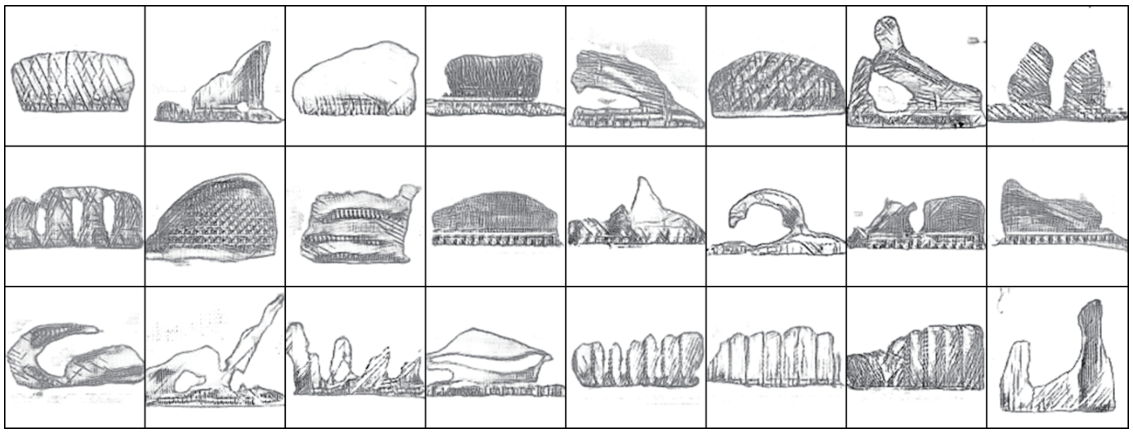

(a) WGAN-GP

Fig. 8 Generated architectural shape sketches by Self-Sparse GAN and WGAN-GP.

(to be continued)

(Continued)



(b) Self-Sparse GAN

Fig. 8 Generated architectural shape sketches by Self-Sparse GAN and WGAN-GP.

Table 10 Comparison between WGAN-GP, SAGAN and Self-Sparse GAN

Dataset	Model	FID
CELEBA-HQ	WGAN-GP	32.40 ± 2.03
	SAGAN	30.94 ± 1.01
	Self-Sparse GAN	**27.72 ± 1.54**
Sketch	WGAN-GP	76.68 ± 1.08
	SAGAN	76.97 ± 1.07
	Self-Sparse GAN	**67.58 ± 0.89**

GP and SAGAN.

4 Conclusion

In this study, a self-sparse generative adversarial network (Self-Sparse GAN) is proposed for the unsupervised image generation task. By exploiting channel sparsity and position sparsity in multi-channel feature maps, Self-Sparse GAN stabilizes the training process and improves the model performance by (1) reducing the search space of convolution parameters in the generator; (2) maintaining meaningful features in the BN layer to alleviate the zero gradient problem; and (3) driving the convolutional weights away from being negative. We demonstrate the proposed method on seven image datasets. Experimental results show that our approach can obtain better FIDs on all the seven datasets compared with WGAN-GP, and is robust to both training hyperparameters and network architectures. In addition, a positive correlation between sparsity and FID further validates that the proposed sparsity module enhances the image generation power of the model. Meanwhile, the proposed method also has the ability to perform the intelligent design of architectural sketches.

Appendix

A MNIST and CIFAR-10

In the following parts, SASTM denotes the self-adaptive sparse transform module. Conv, DeConv, ReLU, and Tanh represent the convolutional layer, deconvolutional layer, rectified linear activation function, and Tanh activation function, respectively. BN-256 and IN-256 represent the input of the batch normalization and instance normalization layer with 256 channels, respectively.

Table A1 Self-Sparse GAN generator and discriminator with 32 × 32 resolution. $z \in \mathbb{R}^{100} \sim N(0,1)$ on MNIST and $z \in \mathbb{R}^{128} \sim N(0,1)$ on CIFAR-10.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-256, ReLU,	Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM** , BN-128, ReLU,	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-64, ReLU,	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv. 4×4, stride=1, padding=0

Table A2 Self-Sparse GAN generator and discriminator with 64 × 64 resolution, $z \in \mathbb{R}^{100} \sim N(0,1)$ on MNIST and $z \in \mathbb{R}^{128} \sim N(0,1)$ on CIFAR-10.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv.4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv.4×4, stride=1, padding=0

Table A3 Self-Sparse GAN generator and discriminator with 128 × 128 resolution, $z \in \mathbb{R}^{100} \sim N(0,1)$ on MNIST and $z \in \mathbb{R}^{128} \sim N(0,1)$ on CIFAR-10.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-512, ReLU	Conv. 4×4, stride=2, padding=1, IN-32, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-256, ReLU	Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-64, ReLU	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-32, ReLU	Conv. 4×4, stride=2, padding=1, IN-512, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv.4×4, stride=1, padding=0

Table A4 SASTM architecture, $z \in \mathbb{R}^{100} \sim N(0,1)$ on MNIST and $z \in \mathbb{R}^{128} \sim N(0,1)$ on CIFAR-10.

Layers	Operators
f	Linear: $\boldsymbol{z} \to 1024$, ReLU Linear: $1024 \to 512$, ReLU
g_1	Linear: $512 \to M_c$, ReLU
g_2	Linear: $512 \to 256$, ReLU Conv or DeConv: $16 \times 16 \to M_h \times M_w$, ReLU

B Fashion-MNIST

Table B1 Self-Sparse GAN generator and discriminator with 64 × 64 × 1 resolution, and $z \in \mathbb{R}^{100} \sim N(0,1)$.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv.4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv.4×4, stride=1, padding=0

Table B2 Self-Sparse GAN generator and discriminator with 128 × 128 × 1 resolution, and $z \in \mathbb{R}^{100} \sim N(0,1)$.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv.4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv. 4×4, stride=1, padding=0

Table B3 SASTM architecture with 64 × 64 × 1 and 128 × 128 × 1 resolution.

Layers	Operators
f	Linear: $\boldsymbol{z} \in \mathbb{R}^{100} \sim N(0,1) \to 1024$, ReLU Linear: $1024 \to 512$, ReLU Linear:$512 \to 256$, ReLU
g_1	Linear: $256 \to M_c$, ReLU
g_2	Conv or DeConv: $16 \times 16 \to M_h \times M_w$, ReLU

C STL-10, CELEBA-HQ, and LSUN bedrooms

Table C1 Self-Sparse GAN generator and discriminator with 64 × 64 × 3 resolution, and $z \in \mathbb{R}^{128} \sim N(0,1)$.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-512, ReLU	Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-256, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-64, ReLU	Conv. 4×4, stride=2, padding=1, IN-512, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv.4×4, stride=1, padding=0

Table C2 Self-Sparse GAN generator and discriminator with 128 × 128 × 3 resolution, and $z \in \mathbb{R}^{128} \sim N(0,1)$.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-512, ReLU	Conv. 4×4, stride=2, padding=1, IN-32, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-256, ReLU	Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-64, ReLU	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-32, ReLU	Conv. 4×4, stride=2, padding=1, IN-512, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv. 4×4, stride=1, padding=0

Table C3 SASTM architecture with 64×64×3 and 128×128 × 3 resolutions.

Layer	Operator
f	Linear: $\boldsymbol{z} \in \mathbb{R}^{128} \sim N(0,1) \to 256$, ReLU
g_1	Linear: $256 \to M_c$, ReLU
g_2	Conv or DeConv: $16 \times 16 \to M_h \times M_w$, ReLU

D Mini-ImageNet

Table D1 Self-Sparse GAN generator and discriminator with 32 × 32 × 3 resolution.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-512, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-256, ReLU	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-512, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv. 4×4, stride=1, padding=0

Table D2 Self-Sparse GAN generator and discriminator with 64 × 64 × 3 resolution.

Generator	Discriminator
DeConv. 4×4, stride=1, padding=0, **SASTM**, BN-512, ReLU	Conv. 4×4, stride=2, padding=1, IN-64, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-256, ReLU	Conv. 4×4, stride=2, padding=1, IN-128, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-128, ReLU	Conv. 4×4, stride=2, padding=1, IN-256, LeakyReLU
DeConv. 4×4, stride=2, padding=1, **SASTM**, BN-64, ReLU	Conv. 4×4, stride=2, padding=1, IN-512, LeakyReLU
DeConv. 4×4, stride=2, padding=1, Tanh	Conv.4×4, stride=1, padding=0

Table D3 SASTM architecture with 32 × 32 × 3 resolution.

Layer	Operator
f	Linear: $\boldsymbol{z} \in \mathbb{R}^{128} \sim N(0,1) \to 1024$, ReLU Linear: $1024 \to 512$, ReLU Linear: $512 \to 256$, ReLU
g_1	Linear: $512 \to M_c$, ReLU
g_2	Linear: $512 \to 256$, ReLU Conv or DeConv: $16 \times 16 \to M_h \times M_w$, ReLU

Table D4 SASTM architecture with 64 × 64 × 3 resolution.

Layer	Operator
f	Linear: $z \in \mathbb{R}^{128} \sim N(0,1) \to 256$, ReLU
g_1	Linear: $256 \to M_c$, ReLU
g_2	Conv or DeConv: $16 \times 16 \to M_h \times M_w$, ReLU

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 51921006 and 52008138), and Heilongjiang Touyan Innovation Team Program (No. AUEA5640200320).

Dates

Received: 26 May 2022; Revised: 7 August 2022; Accepted: 12 August 2022

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proc. 28th Annu. Conf. Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 2672–2680.

[2] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv: 1511.06434, 2015.

[3] L. Mescheder, A. Geiger, and S. Nowozin, Which Training Methods for GANs do actually converge, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 3481–3490.

[4] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, Improved techniques for training GANs, in *Proc. 30th Annu. Conf. Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 2226–2234.

[5] M. Arjovsky and L. Bottou, Towards principled methods for training generative adversarial networks, in *Proc. 5th Int. Conf. Learning Representations*, Toulon, France, 2017.

[6] S. Jenni and P. Favaro, On stabilizing generative adversarial training with noise, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 12137–12145.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, Progressive growing of GANs for improved quality, stability, and variation, arXiv preprint arXiv: 1710.10196, 2017.

[8] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, Self-attention generative adversarial networks, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 7354–7363.

[9] M. Arjovsky, S. Chintala, and L. Bottou, Wasserstein GAN, arXiv preprint arXiv: 1701.07875, 2017.

[10] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, Least squares generative adversarial networks, in *Proc. 2017 IEEE Int. Conf. on Computer Vision*, Venice, Italy, 2017, pp. 2813–2821.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, Improved training of Wasserstein GANs, in *Proc. 31st Annu. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 5767–5777.

[12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, Spectral normalization for generative adversarial networks, in *Proc. 6th Int. Conf. Learning Representations*, Vancouver, Canada, 2018.

[13] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, Sparse convolutional neural networks, in *Proc. 2015 IEEE Conf. Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 806–814.

[14] C. Louizos, M. Welling, and D. P. Kingma, Learning sparse neural networks through L_0 regularization, in *Proc. 6th Int. Conf. Learning Representations*, Vancouver, Canada, 2018.

[15] L. Deng, The MNIST database of handwritten digit images for machine learning research [best of the web], *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[16] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv: 1708.07747, 2017.

[17] A. Krizhevsky, Learning multiple layers of features from tiny images, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[18] A. Coates, A. Ng, and H. Lee, An analysis of single-layer networks in unsupervised feature learning, in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2011, 215–223.

[19] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning, in *Proc. 30th Annu. Conf. Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 3630–3638.

[20] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, LSUN: Construction of a large-scale image dataset using deep learning with humans in the Loop, arXiv preprint arXiv: 1506.03365, 2015

[21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, GANs trained by a two time-scale update rule converge to a local Nash equilibrium, in *Proc. 31st Annu. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 6626–6637.

[22] A. Brock, J. Donahue, and K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, arXiv preprint arXiv: 1809.11096, 2018.

[23] Y. Zhou and T. L. Berg, Learning temporal transformations from time-lapse videos, in *Proc. 14th Eur. Conf. Computer Vision*, Amsterdam, The Netherlands, 2016, pp. 262–277.

[24] P. Isola, J. Y. Zhu, T. Zhou, and Alexei A. Efros, Image-to-image translation with conditional adversarial networks, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 5967–5976.

[25] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, Deblurgan: Blind motion deblurring using conditional adversarial networks, in *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 8183–8192.

[26] M. Huh, S. H. Sun, and N. Zhang, Feedback adversarial learning: Spatial feedback for improving generative adversarial networks, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 1476–1485.

[27] T. Karras, S. Laine, and T. Aila, A style-based generator architecture for generative adversarial networks, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 4396–4405.

[28] T. Chen, M. Lucic, N. Houlsby, and S. Gelly, On self modulation for generative adversarial networks, arXiv preprint arXiv: 1810.01365, 2018.

[29] S. Mahdizadehaghdam, A. Panahi, and H. Krim, Sparse generative adversarial network, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision Workshop*, Seoul, Republic of Korea, 2019, pp. 3063-3071.

[30] F. Liu, L. Jiao, and X. Tang, Task-oriented GAN for PolSAR image classification and clustering, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2707–2719, 2019.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proc. 26th Annu. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

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

[33] R. Shang, W. Zhang, M. Lu, L. Jiao, and Y. Li, Feature selection based on non-negative spectral feature learning and adaptive rank constraint, *Knowl. -Based Syst.*, vol. 236, p. 107749, 2022.

[34] R. Shang, X. Zhang, J. Feng, Y. Li, and L. Jiao, Sparse and low-dimensional representation with maximum entropy adaptive graph for feature selection, *Neurocomputing*, vol. 485, pp. 57–73, 2022.

[35] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang and, H. Lu, Dual attention network for scene segmentation, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 3141–3149.

[36] Y. Wang, Z. Chen, F. Wu, and G. Wang, Person re-identification with cascaded pairwise convolutions, in *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2019, pp. 1470-1478.

[37] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980, 2014.

[38] W. Qian, Y. Xu, and H. Li, A self-sparse generative adversarial network for autonomous early-stage design of architectural sketches, *Comput. -Aided Civ. Infrastruct. Eng.*, vol. 37, no. 5, pp. 612–628, 2021.