Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **11**, Number **1**, February **2026**

Optimization of Generative Adversarial Network Based on Game Theory

Cheng-Qiong Ye[1], Li-Qin Liu[1,*]

[1]School of Big Data and Artificial Intelligence,
Anhui Xinhua University, Hefei 230088, P. R. China
25539948@qq.com, 416049872@qq.com

Yun-Fei Zhu[2]

[2]Thailand Northwestern University, Chiang Mai 20260, Thailand
1468059575@qq.com

[*]Corresponding author: Li-Qin Liu

Received September 20, 2024, revised December 22, 2024, accepted March 07, 2025.

Abstract. *Recently, the research of Generative Adversarial Network (GAN) has made great progress, however, GAN often fails to map input noise to real data distribution, resulting in unstable training and pattern collapse in generating results, which constrain the further application of GAN. Therefore, this paper suggests an optimization model for GAN based on game theory (ETGAN). Firstly, the idea of diffusion model is borrowed, and the forward noise addition method of non-Markov process is introduced to improve the noise addition speed of the diffusion model; the Gaussian distribution is used to implicitly model the reverse denoising process, which improves the inference learning speed of the model, and then improves the stability of GAN training. On this basis, the GAN game process is optimized based on the dynamic game of incomplete information, where different feature generation scenarios on the object are discriminated, constituting a subgame between the generator and the discriminator in the generation process. Secondly, the discriminator extracts the features by calling the encoder in the generator twice. The whole game is actually completed by two parts of the generator. Comparison experiments with cutting-edge GANs models on Cifar-10 and CelebA datasets show that ETGAN improves the IS and FID metrics by 0.4-2.5, thus fully demonstrating that ETGAN can effectively improve the model fitting ability without causing pattern collapse.*
Keywords: Generating adversarial network; Game theory; Pattern collapse; Diffusion model; Gaussian distribution.

1. **Introduction.** Generative Adversarial Network (GAN) is a novel generative model inspired by the idea of zero-sum game. It consists of a generative network and a discriminative network, which learn by letting the two neural networks play against each other, and finally reach the Nash equilibrium [1, 2]. As an important deep learning algorithm, GAN has significant advantages such as computational complexity linearly related to the dimension of the input data, no need to make any a priori assumptions about the target data distribution, and excellent ability to generate realistic samples, and has been widely used in many fields such as computer vision and natural language processing [3, 4, 5]. However, the issues of GAN such as unstable training and pattern collapse in the generated results constrain its further development and application. Therefore, the study of improving GAN and its application can not only improve the generation of various types of image and natural language data, but also inspire and promote various types of learning tasks, which is of great significance for the further development of deep learning models.

1.1. **Related work.** As an implicit generative model based on game theory, GAN generates samples by treating the model as a game between a generative network and a discriminative network, as opposed to explicitly modelling the likelihood function. Choi and Han [6] improve the model performance by aggregating many discriminators to form a composite discriminator, but their multiple models and complex training process make the training cost much higher. Wu et al. [7] used a different classifier from the original model which consisted of a self-encoder whose loss distribution was matched by a loss based on the Wasserstein distance. Nobari et al. [8] proposed the CGAN model by adding the same condition to both the generator and the discriminator and using that condition to determine the direction of the generated model. Liu et al. [9] proposed D2GAN, which uses two discriminators to minimize the KL dispersion and suffers from the problem of high computational complexity. Xiao et al. [10] proposed DRAGAN by considering the gradient update of the generator network and the discriminator network as a regret minimization, but the generalization performance is poor.

Nevertheless, these methods tend to make the training of GAN networks difficult. Even with the addition of a regularization term, the effect is not very significant and the training cost is high. Erdmann et al. [11] used the Wasserstein distance instead of the JS dispersion in order to train the GAN model more consistently, but the convergence was slow. Wu et al. [12] used weight trimming to deal with the Lipschitz constraint in traditional GAN [13], but there is a gradient vanishing. Zhao et al. [14] replaced the original with smaller convolution kernel and added a deep convolution kernel residual network to the original model as a way to improve the model performance. The denoising feature matching proposed by Yan et al. [15] uses the correlation of data between multiple batch to enhance the discriminator's ability to discriminate between real and generated data, but it is computationally intensive. Alonso-Monsalve and Whitehead [16] used deep convolution on GAN instead of the original discriminator, but the generated images lacked global consistency.

Yan et al. [17] proposed the MIX+GAN model, which is based on the min-max game strategy and trains several generator and discriminator models using different parameters simultaneously, but its computational cost is greatly increased. Xu et al. [18] replaced the loss function of GAN with a least squares loss function and defined a callback operator to map the generated samples to the shape of the data stream, but there is the problem of pattern collapse. Guo et al. [19] added the denoising loss function to the discriminator network, which enables the discriminator to obtain more feature information that is distributed with the real data, but with a poor generalization performance. Liu et al. [20] and used the conditional constraints between the generator and discriminator and residual network to optimize the GAN network to have better generalization performance. Fatima and Garapati [21] proposed a GAN-based extension to game theory, which improves the performance of the model by allowing the hidden variables in the generator's input variables to generate more adaptive mutual information with the noise variables.

1.2. **Contribution.** This paper focuses on the training instability and pattern collapse problems in the optimization models of existing GANs, and proposes an optimization model for generative adversarial networks based on game theory (ETGAN). The specific research content is as follows.

(1) Intending to the instability issue in GAN training, this paper carries out forward noise addition and reverse denoising for GAN based on the diffusion model. In order to improve the noise addition and denoising efficiency of the diffusion model, a non-Markov process is introduced for forward noise addition, and a Gaussian distribution is used for reverse denoising, so as to achieve the effective elimination of noise.
(2) Intending to the issue of pattern collapse in generative models, the GAN discriminator is improved from the perspective of game theory, and the discriminator is composed of an encoder and a Bayesian classifier by refining the Bayesian Nash equilibrium and changing the discriminator's recognition task to a feature recognition task.
(3) Since the encoder is used as the discriminator, there is no need to iterate new weights, which reduces the learning time. The game characteristics of the combined generator and discriminator improve the payment matrix and optimize the network structure so that the model can resist pattern collapse without increasing the training burden.
(4) Finally, comparison experiments are conducted on the Cifar10 and CelebA datasets, and the IS and FID values of ETGAN are better than those of the comparison models, which fully proves that ETGAN not only improves the generation quality, but also makes the model more stable.

2. Theoretical analysis.

2.1. **Traditional generative adversarial network.** GAN is a deep generative model. As shown in Figure 1, GAN consists of two opposing models: a generator (G) and a discriminator (D) [22]. G is to generate false samples that are as similar as possible to the real data, while D is to discriminate whether the input to D is from the real data or from the false samples generated by G.

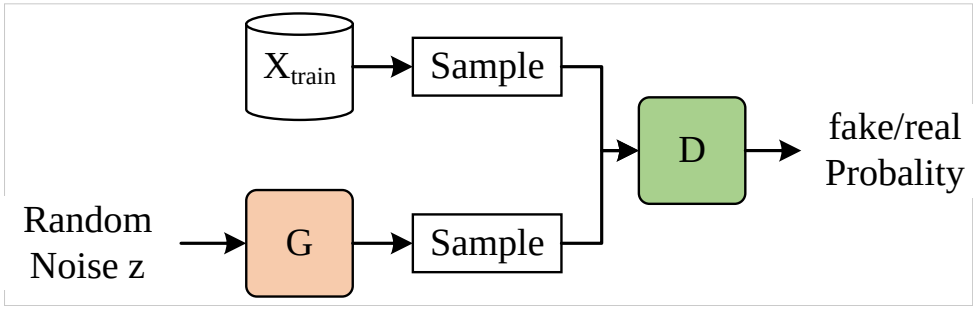

Figure 1. The basic structure of GAN

G randomly draws a noise vector z as input from a predefined noise distribution $p_z(z)$, and then generates a dummy sample $G(z;\theta_g)$, where θ_g contains the parameters of the generated model. Discriminative model $D(x;\theta_d)$ uses true and false samples as inputs for positive and negative samples, respectively, where θ_d contains the parameters of the discriminative model. G and D are trained alternately: while training G, fix D to minimize the loss function $\log(1 - D(G(z)))$ by optimizing θ_g; while training the discriminative model, fix G to minimize the loss function $\log(1 - D(x))$ by optimizing θ_d. That is, G and D are playing a two-player game of minimizing and maximizing the value function $V(G, D)$, i.e., optimizing as follows.

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

Assuming that the distribution function of the real data is $p_{data}(x)$, the distribution function of the fake samples generated by G is $p_g(x)$, then the final G can make $p_g(x) = p_{data}(x)$, i.e., the fake samples can achieve the purpose of the fake to the real.

2.2. **Introduction to Nash equilibrium and game theory.** The main idea of the GAN algorithm comes from the zero-sum game in game theory [23], and the algorithm is trained iteratively so that the two sides of the game converge to a Nash equilibrium. Due to the complexity of GAN and the dynamic adjustment in the training process, it makes it difficult to reach the real Nash equilibrium state in the actual training, resulting in an unstable training process. To cope with the above issues, this paper introduces Refined Bayesian Nash Equilibrium (RBNE) [24] to incorporate a new generative training paradigm without changing the original model as much as possible.

A combination of strategies $s^* = (s_1^*, s_2^*, \ldots, s_n^*) \in S$ is called a pure strategy Nash equilibrium, if $\forall i \in N$, then there is $f_i(s_1^*, \ldots, s_i^*, \ldots, s_n^*) \geq f_i(s_1^*, \ldots, s_i, \ldots, s_n^*)$ where $\forall s_i \in S_i$. The set of participants is $N = \{1, 2, \ldots, n\}$, S_i is the set of strategies of participant i, S denotes the Cartesian product of the sets of strategies of all participants, $S = \prod_{j=1}^{n} S_j$, and the payoff function of participant i is f_i, $f_i : S \to R$, $\forall i \in N$. A game is called a finite game if $\forall i \in N, S_i$ are finite sets.

The traditional game assumes that each participant is fully aware of the other participants' information, which is often difficult to achieve in reality. Therefore, scholars have proposed incomplete information games, in which participants do not have accurate information about the others' characteristics, strategy spaces and payoff functions. Incomplete information games can be divided into incomplete information static games and incomplete information dynamic games [25], and this paper mainly deals with the RBNE in incomplete information dynamic games. The RBNE can be expressed as a combination of strategies: $s^*(\theta) = (s_1^*(\theta_1), \ldots, s_n^*(\theta_n))$ and a posteriori probability combination $p = (p_1, p_2, \ldots, p_n)$ satisfying the following two conditions.

(1) For all participants i, at each information set h, there is Equation (2).

$$s_i^*(s_{-i}, \theta_i) \in \arg\max_{s_i} \sum_{\theta_{-i}} p_i(\theta_{-i}|a_{-i}^h) u_i(s_i, s_{-i}, \theta_i) \tag{2}$$

(2) $p_i(\theta_i|a_{-i}^h)$ is obtained from the prior probability $p_i(\theta_{-i}|\theta_i)$, the observed α_{-i}^h and the optimal strategy $s_{-i}^*(\cdot)$ using Bayes' Rule.

3. Forward noise addition and backward denoising for GAN based on diffusion models.

3.1. **Forward noise addition process.** How to solve the instability of GAN in the training process is a very tough issue in the field of generative learning. As an effective generative model [26], the core component of the diffusion model is the conditional diffusion process, in which the noisy data gradually becomes clearer after several diffusion steps, and finally an approximation of the original data is generated. Compared with GAN, diffusion models show higher stability during training. Therefore, combining the diffusion model with GAN becomes an effective way to address the instability of GAN training.

Adding Gaussian noise to the original object x during data processing increases both the stochasticity of the data and the tolerance of the model to noise, thus enhancing the robustness and generalization of the model. Traditional GAN uses a fixed Markov chain for forward noise addition but cannot dynamically adjust the intensity and distribution of noise according to the characteristics of the object, so this paper invokes the non-Markov process noise addition method for flexible forward noise addition.

The diffusion model is denoted as $p_\theta(x_0) := \int p_\theta(x_{0:T})dx_{1:T}$, $x_1, \ldots, x_T$ is a hidden variable with the same dimension as the data x_0, $p(x_0)$ is the original data distribution with initial state $p(x_T) = N(x_T; 0, I)$, $p(x_{1:T}|x_0)$ is the approximate posterior distribution of the diffusion model, whose expression can be expressed as follows.

$$p(x_{1:T}|x_0) := p(x_T|x_0)\prod_{t=2}^{T} p(x_{t-1}|x_t, x_0) \tag{3}$$

where $p(x_T|x_0) = N(\sqrt{\bar{\alpha}_T}x_0, (1-\bar{\alpha}_T)I)$ holds when $t > 1$.

According to Bayes' theorem [27], the true denoising distribution $p(x_{t-1}|x_t, x_0)$ of the diffusion model can be introduced as follows.

$$p(x_{t-1}|x_t, x_0) = N\left(\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2 I\right) \tag{4}$$

where α_t denotes the observed data at moment t and σ denotes the variance.

3.2. **Reverse denoising process.** In this section, the denoising distribution is modelled using a Gaussian distribution, assuming a small value of T, which implies that each diffusion step has a large β_t value. To relate this diffusion process to GAN, a measure of the gap between the generator distribution $p_\theta(x_{t-1}|x_t)$ and the true denoising distribution $p(x_{t-1}|x_t)$ is introduced, i.e., by calculating the scatter between the two, as shown in Equation (5).

$$\min_\theta \sum_{t\geq 1} E_{p(x_t)}[D_{adv}(p(x_{t-1}|x_t), p_\theta(x_{t-1}|x_t))] \tag{5}$$

where the Wasserstein distance is used to measure the gap between the generator distribution $p_\theta(x_{t-1}|x_t)$ and the true denoising distribution $p(x_{t-1}|x_t)$. To establish adversarial training, the discriminator (D) can be obtained by the above forward noise addition process as follows.

$$D_\phi(x_{t-1}, x_t, t) : R^N \times R^N \times R \to [0, 1] \tag{6}$$

where ϕ is a parameter of D, x_{t-1} and x_t are inputs to D.

Instead of predicting x_{t-1} directly in the denoising step, the diffusion model uses parametric implicit modelling of the backward denoising process given D. Instead, a parametric denoising model $p_\theta(x_{t-1}|x_t)$ is built first.

$$p_\theta(x_{t-1}|x_t) := p(x_{t-1}|x_t, x_0 = f_\theta(x_t, t)) \tag{7}$$

In Equation (7), x_0 is first predicted by the implicit denoising model $f_\theta(x_t, t)$, and then x_{t-1} is sampled by the posterior distribution $p(x_{t-1}|x_t, x_0)$ given x_t and the prediction x_0. Intuitively, $p(x_{t-1}|x_t, x_0)$ is the distribution on x_{t-1}, which always has the form of a Gaussian distribution when performing backward denoising from x_t to x_0. By adding hidden variables to the parametric denoising model $p_\theta(x_{t-1}|x_t)$, Equation (7) is further written as follows.

$$\begin{aligned} p_\theta(x_{t-1}|x_t) &= \int p_\theta(x_0|x_t)p(x_{t-1}|x_t, x_0)dx_0 \\ &= \int p(z)p(x_{t-1}|x_t, x_0 = G_\theta(x_t, z, t))dz \end{aligned} \tag{8}$$

where z is the hidden variable, $z \sim p(z) := N(z; 0, I)$.

4. Optimization of GAN based on game theory.

4.1. Game theory based GAN generator design.

After the forward and reverse denoising of D and G in GAN, in order to solve the pattern collapse problem of GAN during the training process, this paper further optimizes GAN based on game theory (ETGAN). Since part of the function of D is realized by the encoder, self-updating is performed through the game. D converges to the Nash equilibrium point when the GAN training is completed converts to solving the RBNE in the incomplete information dynamic game. the new GAN framework does not have to worry about the training collapse and falling into the local Nash equilibrium problem, and interacts with a certain frequency during the training process to achieve the optimal global solution, the specific algorithmic flow is shown in Figure 2.

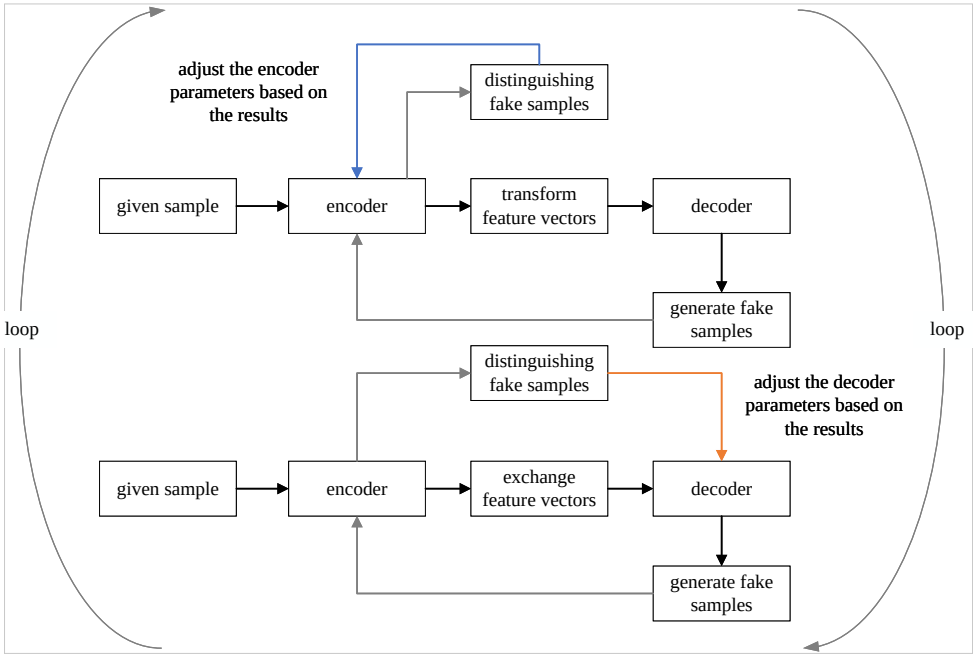

Figure 2. Optimization process of GAN model

G can be regarded as a U-Net system [28] composed of decoder and encoder to complete the generation task, which is different from the traditional GAN in that G only performs the training of decoder weights. The encoder is then trained in D. During the operation of the algorithm the encoder extracts the required feature vectors, the decoder encodes the required feature vectors into the object containing the formulated feature, the original object A containing the feature, and the object B not containing the feature. The corresponding obscured level codes in A and B will be extracted by the encoder, and the corresponding codes containing n features will be obtained.

$$\begin{cases} Axs = [a_1, a_2, a_3, \ldots, a_i, \ldots, a_n] \\ Bes = [b_1, b_2, b_3, \ldots, b_i, \ldots, b_n] \end{cases} \tag{9}$$

where a_n, b_n represent the n feature vectors obtained by the encoder, since A contains the specified feature and B does not contain the specified feature, the presence or absence of the feature is abbreviated as $label(A) = [u_1, u_2, u_3, \ldots, 1, \ldots, u_n]$, $label(B) = [v_1, v_2, v_3, \ldots, 0, \ldots, v_n]$ using labels, which are coded as 0-1 to indicate whether or not the i-th feature is contained.

After obtaining Axs and Bes, the decoder performs a reduction operation to check if there are too many differences between the decoded reconstruction and the original object. Make sure that the encoder can return to the original space by inverse mapping. Secondly, the copying operation is carried out in order to eliminate the overfitting effect caused by the zeroing operation, noting that the object after copying of A is C, and the object after copying of B is D. Then, the i-th code in the hidden coding of C and D is exchanged, as shown in Equation (10).

$$\begin{cases} Cxs = [a_1, a_2, a_3, \ldots, a_i, \ldots, a_n] \\ Des = [b_1, b_2, b_3, \ldots, b_i, \ldots, b_n] \\ Ces = [a_1, a_2, a_3, \ldots, b_i, \ldots, a_n] \\ Dxs = [b_1, b_2, b_3, \ldots, a_i, \ldots, b_n] \end{cases} \tag{10}$$

To realize the conversion operation of the obscured level encoding, the decoder first links the obscured level encoding through the intermediate level to ensure the consistency between the input and the output of the encoder, and then realizes the generation of the object through the encoding link, as shown in Equation (11), and the structure of G is shown in Figure 3.

$$\begin{cases} label(dec(Axs)) = label(A^{reconsitutio}) = label(A) \\ label(dec(Bes)) = label(B^{reconsitutio}) = label(B) \\ label(dec(Ces)) = [u_1, u_2, u_3, \ldots, 0, \ldots, u_n] = label(C) \\ label(dec(Dxs)) = [v_1, v_2, v_3, \ldots, 1, \ldots, v_n] = label(D) \end{cases} \tag{11}$$

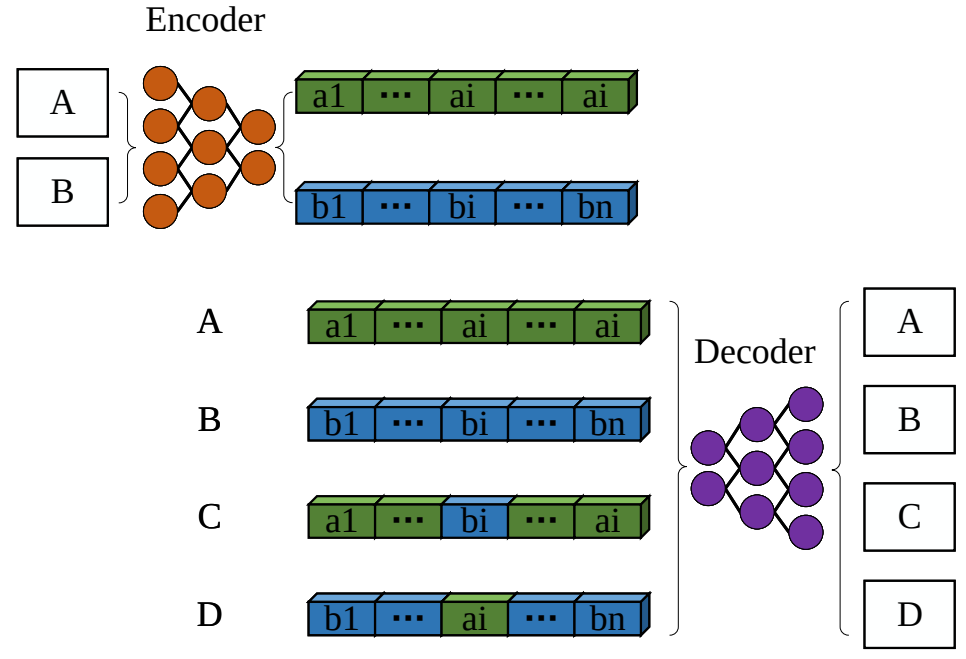

Figure 3. Generator structure in ETGAN

G is mainly implemented by controlling the decoder weights, so the set of strategies of G in this game is the weight matrix in the hidden layer encoding $\alpha_{iwhc} \in [-1, 1]$. The purpose of G is to generate pseudo-objects that can be the original objects, and to enhance the misjudgement of the pseudo-objects by D. Thus, the payoff function of G in this game is as follows.

$$e_i^*(s_{yes}, x_{fake}) \in \arg \max_{label_d(x), x_{real}} \sum D(x_{fake}|\alpha_i) U(label_d(x), s_{yes}, x_{fake}) \tag{12}$$

where s_{yes} is D's strategy for selecting 'true samples' and x_{real} is for reading 'false samples', i.e., generating samples, $U(label_d(x), s_{yes}, x_{fake})$ is the utility function corresponding to G.

$$U(label_d(x), s_{yes}, x_{fake}) = \sum_{i=1}^{n} u_i \tag{13}$$

where $u_1, u_2, \ldots, u_n$ is coded 0 - 1, representing whether D can be mixed into the original object for different features of the generated object.

4.2. **Game theory based GAN discriminator design.** D identifies the different feature codes generated by Bayesian classifier. From the perspective of game theory, D obtains the identification result by 'observing' the input object after G has made a strategic choice. Since D is different from the decoder in G, it cannot share the parameter information with the decoder, and thus D does not know all the information of G. The two parties constitute an incomplete information dynamic game, and the structure of D is shown in Figure 4.

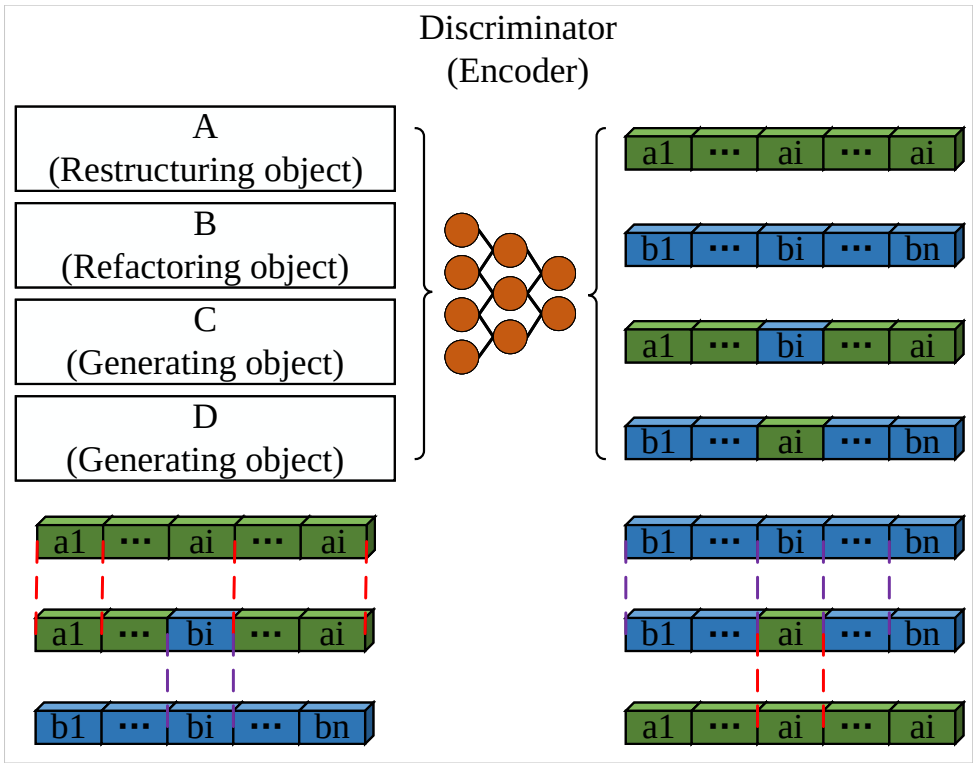

Figure 4. Discriminator structure in ETGAN

D obtains the input object feature code through the encoder, and for the i-th feature a_i, the prior probability $D(\alpha_i|X)$ is obtained, and the sample x input to D is divided into true sample x_{real} and false sample x_{fake}. The total sample $X = x_{real} + x_{fake}$, the posterior probability $D(x_{real}|\alpha_i), D(x_{fake}|\alpha_i)$ is found by Bayes' rule.

$$D(x_{real}|\alpha_i) = \frac{D(\alpha_i|x_{real})D(x_{real})}{D(x_{real})D(\alpha_i|x_{real}) + D(x_{fake})D(\alpha_i|x_{fake})} \tag{14}$$

$$D(x_{fake}|\alpha_i) = \frac{D(\alpha_i|x_{fake})D(x_{fake})}{D(x_{fake})D(\alpha_i|x_{fake}) + D(x_{real})D(\alpha_i|x_{real})} \tag{15}$$

where $D(x_{real})$ and $D(x_{fake})$ are the proportion of true samples and false samples input to the algorithm respectively.

According to the obtained a posteriori probability of each feature scoring, using 0-1 coding implementation, 0 means that the judgement and the original sample features are too different, 1 means that the recognition of the feature code and the original sample is similar to the judgement results of different features constitute the D strategy to choose $label_d(x) = [u_1, u_2, \ldots, u_n]$, where $u_1, u_2, \ldots, u_n$ are 0-1 coding, x is the input object. After obtaining the discriminant result, the appropriate utility function is constructed as follows.

$$U(label_d(x), s_{Builder}, x_{real}) = \sum_{i=1}^{n} u_i \tag{16}$$

where $s_{Builder}$ is the strategy chosen by G, the RBNE searched by D is a combination of $U(label_d(x), s_{Builder}, x_{real})$ and $D(x_{real}|\alpha_i)$, denoted as $s_i^*(s_{Builder}, x_{real})$, and the refinement process allows $s_i^*(s_{Builder}, x_{real})$ to reach its maximum value.

$$s_i^*(s_{Builder}, x_{real}) \in \arg \max_{label_d(x), x_{real}} \sum D(x_{real}|\alpha_i) U(label_d(x), s_{Builder}, x_{real}) \tag{17}$$

As the number of iterations increases, D reaches the Nash equilibrium of the partial subgame on some features, and when the encoder can separate all the features efficiently, D converges to RBNE. At this point, we stop updating the parameters until G cannot converge to RBNE after iterative training and restart the iterative training.

4.3. **Loss function design.** The loss function of the algorithm is determined by the payoff functions of the two sides of the game, and the loss function of the whole GAN contains the loss functions of D and G, which are alternately optimized to make the model converge eventually. The loss function is composed in the form of a probability, and the loss function of D is divided into two parts: the first part is the reconstruction loss function, which ensures that the difference between the generated object and the original object is not too large.

$$loss_{rec} = ||A - A^{re}|| - ||B - B^{re}|| \tag{18}$$

As the decoder and encoder update their weights with training, the generated object gets closer and closer to the real object, and $loss_{rec}$ gets closer and closer to zero. The other part is the adversarial loss function that discriminates the encoder from G as follows.

$$loss_{gd} = -E(\log(D_1(label(B)|C)D_1(C))) - E(\log(D_2(label(A)|C)D_1(C))) \tag{19}$$

where $D_1(label(B)|C)$ indicates that the switched feature code of the original object $D_2(label(A)|C)$ can be read on the generated object C, and $loss_{gd}$ indicates that the unswitched feature code of the original object A can be read on C. In order to minimize $loss_{gd}$, $D_1(label(B)|C)$ and $D_2(label(A)|C)$ in Equation (19) must be maximized as much as possible. The more real the generated object is, the smaller $loss_{gd}$ is. And for G, the total loss function is as follows.

$$loss_g = loss_{gd} + loss_{rec} \tag{20}$$

D is also composed of two parts, the first part is the loss function of the encoder to identify the selected exchange features, as shown in Equation (21).

$$\begin{aligned} loss_s = &- E(\log(D_1(label(A)|A))) - E(\log(1 - D_1(label(B)|C))) \\ &- E(\log(D_1(label(B)|B))) - E(\log(1 - D_1(label(A)|D))) \end{aligned} \tag{21}$$

The second part is the loss function for the encoder to extract the discrimination of the non-selected exchange features as in Equation (22).

$$\begin{aligned} loss_i = &- E(\log(D_2(label(A)|A))) - E(\log(1 - D_2(label(A)|C))) \\ = &-E(\log(D_2(label(B)|B))) - E(\log(1 - D_2(label(B)|D))) \end{aligned} \tag{22}$$

Structurally, D_1 and D_2 are the same encoder, only the discrimination intervals are different. The total loss function is as follows.

$$loss_d = loss_s + loss_i \tag{23}$$

To make the loss function small enough is to minimize $D_1(label(A)|C)$ and $D_1(label(B)|D)$ together, and to maximize $D_1(label(A)|A)$ and $D_1(label(B)|B)$. The same is true for D_2, so that D is written with the correct labels as much as possible, increasing the saliency of each feature of the G-generated object.

5. Experiments and analysis of results.

5.1. **Optimization performance comparison of GAN.** In this paper, two datasets, Cifar10 and CelebA, are selected to validate the ETGAN model in this paper. The Cifar10 dataset provides 60,000 color images of 32*32 pixels in size, divided into 10 classes, with 6,000 images in each class, which is an open object recognition dataset. CelebA contains a total of 202,599 feature-labelled face images from 10,177 celebrities. The initial learning rate is set to 0.0002. The Cifar10 image is small, so the batch size can be set to 2500, while the CelebA input image is large, the batch size can only be set to 1000. Using the Adam optimizer with a gradient penalty of 10 and a consistency penalty of 2.

The hardware configurations for the experiments are as follows: the CPU is Intel(R) Core i9-9900K@3.60 GHz, the GPU is NVIDIA GeForce RTX 3060, the operating system used for the experiments is Windows 10, and the deep learning frameworks are Pytorch1.8.1, CUDA11.1, and cudnn8.0.4. This experiment

uses Inception Score [29] (IS) and Frechet Inception Distance [30] (FID), which are commonly used in GAN model performance evaluation, to evaluate the pattern collapse and optimization performance of ETGAN, IEGAN [18], FuseGAN [19], ACGAN [20] and DLGAN [21]. pattern collapse and optimization performance. Higher IS scores indicate better model performance, while lower FID scores indicate better model performance.

The training curves of IS and FID for the five models on the Cifar10 dataset are shown in Figures 5 and 6, respectively, and ETGAN achieves the best performance in both metrics. The IS values of ETGAN, FuseGAN, ACGAN and DLGAN converge to 7.6931, 7.3042, 7.2514 and 6.4720 after 100 times of training, and the FID values converge to 24.7202, 30.8164, 28.2539 and 27.1862, respectively, and the IEGAN suffers from a pattern collapse on this data set phenomenon and both IS and FID could not converge. When ETGAN is trained to a certain extent, it is often necessary to increase the strength of information interaction to help game opponents understand each other's strategies to facilitate the actual training. In the later stages of model training, both players tend to reach a higher level of the game, and in the middle of the game there is a jump out of the local game, so that this dynamic interaction prevents both players from falling into a low-level game cycle again, thus improving the optimization performance.

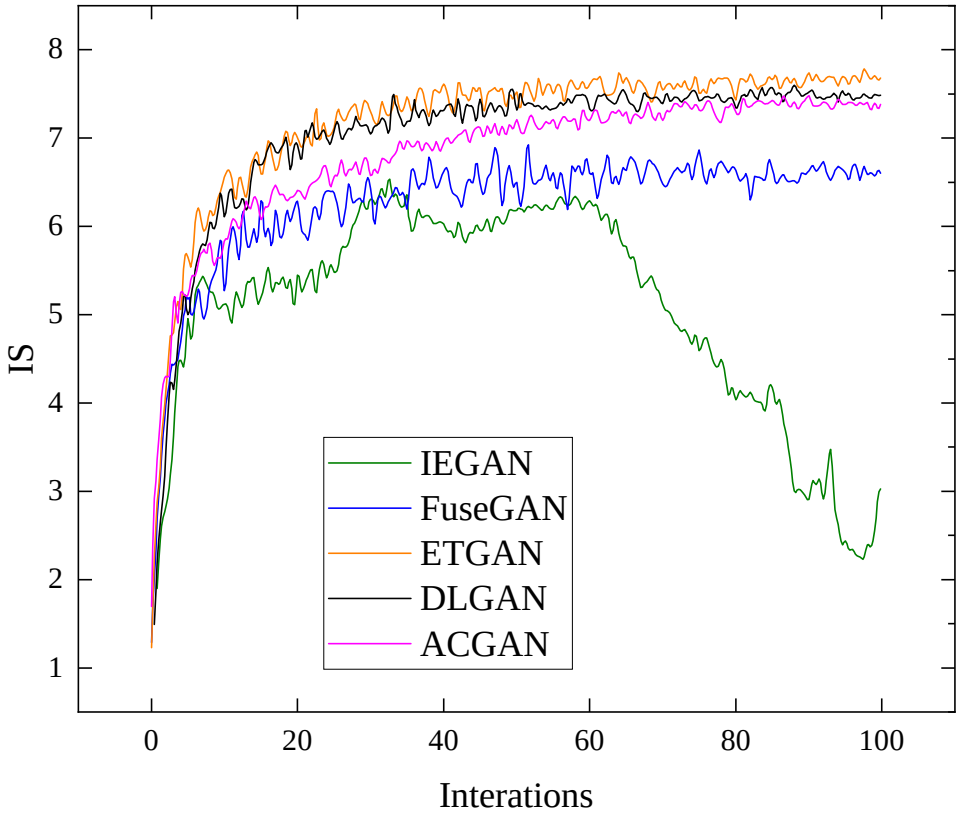

Figure 5. IS value of different GAN optimization models on dataset Cifar10

The best IS and best FID for the five models on the CelebA dataset are shown in Table 1, with the best performing values in both metrics bolded. ETGAN performs best on IS and FID, which is consistent with the analyses on the Cifar10 dataset, further demonstrating the superiority of ETGAN. The performance of ACGAN and DLGAN is closer, ACGAN is optimized for GAN using residual network and DLGAN is Nash equilibrium by expanding the game relationship between D and G. However, the performance of both is inferior to that of ETGAN. IEGAN replaces the traditional loss function by a least squares loss function, but training is slow and leads to pattern collapse. The presence of redundant or counterproductive network layers in FuseGAN leads to poor results after optimization. Therefore, ETGAN has excellent performance on both datasets.

5.2. **Generate quality analyses.** In addition to analyzing the performance of the optimized GAN, its generation quality needs to be further evaluated. In this section, the generation quality of different models is compared by peak signal-to-noise ratio (PSNR) and structural similarity of images (SSIM) on Cifar10 and CelebA datasets, as shown in Figure 7.

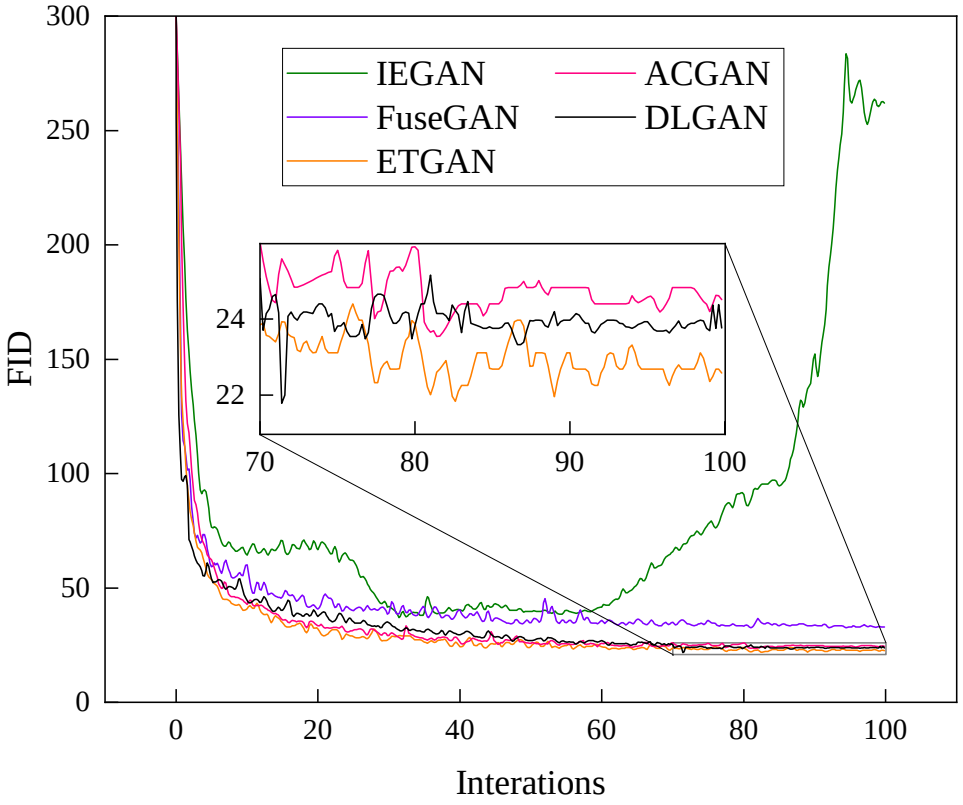

Figure 6. FID value of different GAN optimization models on dataset Cifar10

Table 1. Best IS and best FID for five models on the CelebA dataset

Model	IS↑	FID↓
ETGAN	**7.9647**	**22.1365**
IEGAN	6.4136	37.4481
FuseGAN	7.1381	29.6328
ACGAN	7.4932	26.9614
DLGAN	7.5062	24.1892

The performance of PSNR and SSIM of ETGAN based on dynamic game with incomplete information is better than that of the other four models. In Cifar10 dataset, the PSNR and SSIM of ETGAN are 31.74 dB and 0.952, respectively, which are 11.21 dB and 0.11 dB higher than IEGAN. Compared with FuseGAN, FuseGAN increased by 5.83 dB and 0.059, ACGAN increased by 3.38 dB and 0.024, and DLGAN increased by 2.4 dB and 0.05, respectively. On CelebA dataset, the PSNR and SSIM of ETGAN are 29.66 dB and 0.939, respectively, which are both higher than the comparison model, fully verifying that ETGAN can significantly improve the generation quality of GAN.

6. **Conclusion.** Taking GAN as the starting point, this paper suggests a generative adversarial network optimization model based on game theory (ETGAN), focusing on how to improve the training stability of the model and mitigate the collapse of the model. Firstly, in terms of improving the training stability of GAN, the forward noise and reverse noise denoising of GAN are carried out based on the diffusion model to achieve effective noise elimination and solve the instability problem in GAN training. Secondly, in terms of mitigating pattern collapse, GAN is improved from the incomplete information game. Relied its compilation architecture of the hidden layer encoding, the recognition task of the discriminator is changed to the feature recognition task through RBNE, and the payoff matrix is improved by combining the characteristics of the game of the generator and the discriminator to optimize the structure of the network, which makes the model resistant to pattern collapse without increasing the burden of training.

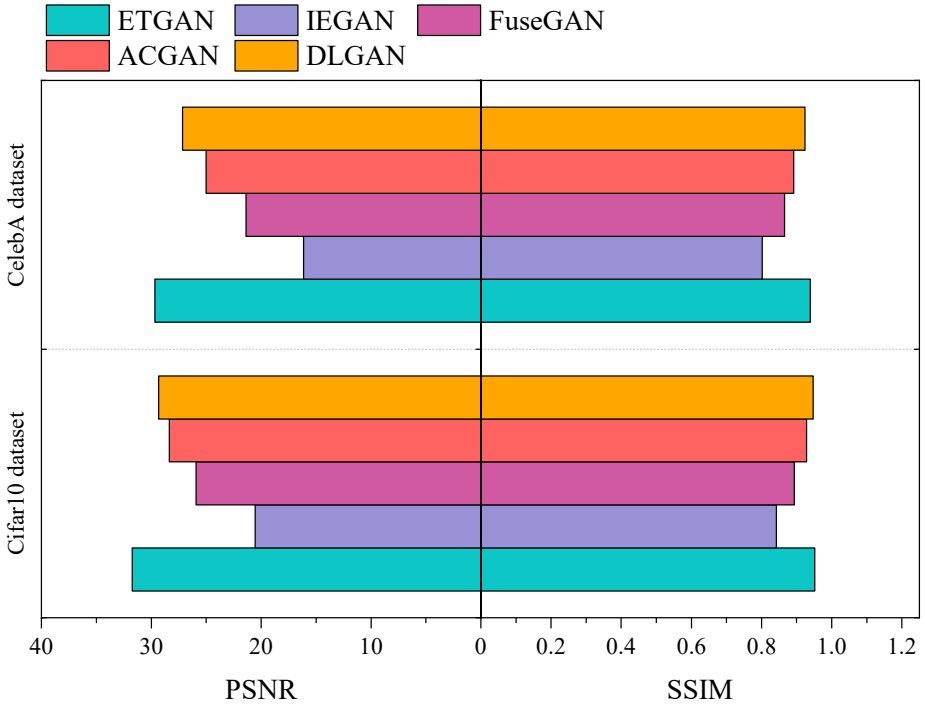

Figure 7. Generation quality of different GAN optimization models

Finally, experiments on the Cifar10 and CelebA datasets verify that ETGAN can converge smoothly without causing pattern collapse and helps to improve the quality of generated content. The design of the generator and discriminator will be further investigated in the future to try to introduce more advanced continuous normalized flow models or other optimization techniques to improve model performance and stability.

Acknowledgment. This work is supported by the Natural science research project of Anhui Xinhua University (No. 2024zr001), the Anhui Province Teaching Demonstration Course: Operating System Principles (No. 2020SJJXSFK1294), and the Anhui Province Quality Engineering Project: Operating System Principles (No. 2019kfkc158).

REFERENCES

[1] M. Mohebbi Moghaddam, B. Boroomand, M. Jalali, A. Zareian, A. Daeijavad, M. H. Manshaei, and M. Krunz, "Games of GANs: Game-theoretical models for generative adversarial networks," *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9771-9807, 2023.

[2] A. S. Chivukula, X. Yang, W. Liu, T. Zhu, and W. Zhou, "Game theoretical adversarial deep learning with variational adversaries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 11, pp. 3568-3581, 2020.

[3] T. Hazra, and K. Anjaria, "Applications of game theory in deep learning: a survey," *Multimedia Tools and Applications*, vol. 81, no. 6, pp. 8963-8994, 2022.

[4] J. Cheng, Y. Yang, X. Tang, N. Xiong, Y. Zhang, and F. Lei, "Generative adversarial networks: A literature review," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 14, no. 12, pp. 4625-4647, 2020.

[5] S.-W. Park, J.-S. Ko, J.-H. Huh, and J.-C. Kim, "Review on generative adversarial networks: focusing on computer vision and its applications," *Electronics*, vol. 10, no. 10, 1216, 2021.

[6] J. Choi, and B. Han, "Mcl-gan: Generative adversarial networks with multiple specialized discriminators," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29597-29609, 2022.

[7] E. Wu, H. Cui, and R. E. Welsch, "Dual autoencoders generative adversarial network for imbalanced classification problem," *IEEE Access*, vol. 8, pp. 91265-91275, 2020.

[8] A. H. Nobari, W. Chen, and F. Ahmed, "Range-constrained generative adversarial network: Design synthesis under constraints using conditional generative adversarial networks," *Journal of Mechanical Design*, vol. 144, no. 2, 021708, 2022.

[9] B. Liu, L. Wang, J. Wang, and J. Zhang, "Dual discriminator weighted mixture generative adversarial network for image generation," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 8, pp. 10013-10025, 2023.
[10] Y. Xiao, C.-M. Pun, and B. Liu, "Adversarial example generation with adaptive gradient search for single and ensemble deep neural network," *Information Sciences*, vol. 528, pp. 147-167, 2020.
[11] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, "Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks," *Computing and Software for Big Science*, vol. 2, pp. 1-9, 2018.
[12] P. Wu, W. Yuan, L. Ji, L. Zhou, Z. Zhou, W. Feng, and Y. Guo, "Missile aerodynamic shape optimization design using deep neural networks," *Aerospace Science and Technology*, vol. 126, 107640, 2022.
[13] K. Liu, and G. Qiu, "Lipschitz constrained GANs via boundedness and continuity," *Neural Computing and Applications*, vol. 32, pp. 18271-18283, 2020.
[14] H. Zhao, D. Wu, H. Su, S. Zheng, and J. Chen, "Gradient-based conditional generative adversarial network for non-uniform blind deblurring via DenseResNet," *Journal of Visual Communication and Image Representation*, vol. 74, 102921, 2021.
[15] L. Yan, J. Fu, C. Wang, Z. Ye, H. Chen, and H. Ling, "Enhanced network optimized generative adversarial network for image enhancement," *Multimedia Tools and Applications*, vol. 80, pp. 14363-14381, 2021.
[16] S. Alonso-Monsalve, and L. H. Whitehead, "Image-based model parameter optimization using model-assisted generative adversarial networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5645-5650, 2020.
[17] N. Feldkamp, S. Bergmann, F. Conrad, and S. Strassburger, "A Method Using Generative Adversarial Networks for Robustness Optimization," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 32, no. 2, pp. 1-22, 2022.
[18] C. Xu, Y. Cui, Y. Zhang, P. Gao, and J. Xu, "Image enhancement algorithm based on generative adversarial network in combination of improved game adversarial loss mechanism," *Multimedia Tools and Applications*, vol. 79, pp. 9435-9450, 2020.
[19] X. Guo, R. Nie, J. Cao, D. Zhou, L. Mei, and K. He, "FuseGAN: Learning to fuse multi-focus image via conditional generative adversarial network," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 1982-1996, 2019.
[20] H. Liu, Z. Hu, J. Yu, and S. Gao, "Conditional GAN-based remote sensing target image generation method," *International Journal of Advanced Network, Monitoring and Controls*, vol. 5, no. 4, pp. 66-74, 2020.
[21] S. I. Fatima, and Y. Garapati, "Generative adversarial deep learning in images using Nash equilibrium game theory," *International Journal of Electrical & Computer Engineering*, vol. 13, no. 6, pp. 6351-6360, 2023.
[22] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1-43, 2019.
[23] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 19-46, 2024.
[24] F. Zhang, T.-Y. Wu, J.-S. Pan, G. Ding, and Z. Li, "Human motion recognition based on SVM in VR art media interaction environment," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, 2019.
[25] Y. Ma, Y. Peng, and T.-Y. Wu, "Transfer learning model for false positive reduction in lymph node detection via sparse coding and deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121-2133, 2022.
[26] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10850-10869, 2023.
[27] M. A. Medow, and C. R. Lucey, "A qualitative approach to Bayes' theorem," *Bmj Evidence-based Medicine*, vol. 16, no. 6, pp. 163-167, 2011.
[28] Y. Enokiya, Y. Iwamoto, Y.-W. Chen, and X.-H. Han, "Automatic liver segmentation using U-Net with Wasserstein GANs," *Journal of Image and Graphics*, vol. 6, no. 2, pp. 152-159, 2018.
[29] Q. Lin, Z. Fang, Y. Chen, K. C. Tan, and Y. Li, "Evolutionary architectural search for generative adversarial networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 4, pp. 783-794, 2022.

[30] L. F. Buzuti, and C. E. Thomaz, "Fréchet AutoEncoder distance: a new approach for evaluation of generative adversarial networks," *Computer Vision and Image Understanding*, vol. 235, 103768, 2023.