





## Research Article

# Waveform Reconstruction of DSSS Signal Based on VAE-GAN

Qi Feng <sup>1</sup>, Junyi Zhang <sup>1,2</sup>, Li Chen <sup>1</sup>, and Fang Liu <sup>1,2</sup>

<sup>1</sup>Signal Intelligence and Electronic Warfare Department, The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, 050011 Hebei, China

<sup>2</sup>Hebei Key Laboratory of Electromagnetic Spectrum Cognition and Control, China

Correspondence should be addressed to Qi Feng; fq\_learning@outlook.com

Received 20 April 2022; Accepted 6 June 2022; Published 4 July 2022

Academic Editor: Mingqian Liu

Copyright © 2022 Qi Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The complex electromagnetic environment will limit the efficacy of communication equipment. It is critical to construct a complex electromagnetic environment to test communication equipment in order to maximize its capability. One of the most important methods for constructing a complex electromagnetic environment is signal reconstruction. This paper proposes a VAE-GAN-based method for reconstructing direct sequence spread spectrum (DSSS) signals. In this method, the deep residual shrinkage network (DRSN) and self-attention mechanism are added to the encoder and discriminator of VAE-GAN. In feature learning, the DRSNs can reduce the redundant information caused by noise in the collected signal. The self-attention mechanism can establish the long-distance dependence between the input sequences, making it easier for the network to learn the samples' pseudonoise (PN) sequence features. In addition, feature loss is applied to the encoder and generator to improve network stability during training. The results of the experiments indicate that this method can reconstruct DSSS signals with the characteristics of the target signal.

## 1. Introduction

The electromagnetic environment of today is quite complex [1]. Communication signals are accompanied by more complicated signals such as jamming, inadvertent crosstalk, and natural thunderstorms [2–4], whether in the air, sea, or land. In order to adapt to the current development of the communication industry, in recent years, machine learning has been widely used to strengthen electromagnetic spectrum management and ensure the reliable transmission of wireless communication [5, 6]. The construction of a complex electromagnetic environment is important for improving the adaptability of its own communication system to a specific electromagnetic environment [7], and the reconstruction of noncooperating parties' communication signals is an important part of constructing a complex electromagnetic environment [8].

The traditional signal reconstruction approach necessitates complicated parameter measurement and analysis of the signal of the noncooperating party [9, 10], which increases the manual decision-making process and costs

time. Furthermore, there are many complex signals with unique structures in today's electromagnetic environment. It is challenging to properly characterize the properties of a target signal using traditional methods, necessitating the development of a new signal reconstruction method. With the development of deep learning, generative adversarial networks [11] (GAN) provide the solution to this challenge. When the features of the target signal are unknown, GAN learns the distribution of samples in space through the competition of two neural networks and generates data that conforms to the target characteristics.

In terms of using GAN to reconstruct signals, predecessors have carried out much research. Qin [12] reconstructed AM and CPFSK signals by using improved CGAN, and the reconstructed signals accord with the characteristics of the sample; Yang et al. [13] used BEGAN to reconstruct BPSK and 8PSK signal waveforms with acceptable quality; Shi et al. [14] used GAN to reconstruct the signal transmitted by the defender, and the resulting QPSK wireless spoofing signal was used to deceive the defender, with a deception rate of 76.2%; Zhao and Jin [15] used a very basic GAN in

their research. The network generated communication interference waveforms such as BPSK, QPSK, 16QAM, and 2FSK, as well as verifying the network's generalization. However, for DSSS signals with more complex structures, general GAN tends to ignore the higher-dimensional features of the PN sequence during the learning process, so no solution has been proposed at present.

This paper presents a method of reconstructing DSSS signals based on VAE-GAN [16]. We add a DRSN [17, 18] and a self-attention mechanism [19] to the encoder and discriminator. The DRSN can denoise the sample signal, reduce the influence of noise on the original characteristics of the signal, and make the network focus on the DSSS signal characteristics during the learning process. In addition, the residual network part can make the model suitable for the training of a variety of DSSS samples with different PN sequence period lengths, which prevents network degradation. The self-attention mechanism can make the network pay attention to the correlation between data points at different positions in the whole input data, making it easier for the model to learn the PN sequence existing in the sample. We also replace the learned similarity part of the loss function of the generator and the encoder with a feature loss, which is more suitable for measuring signals containing PN sequences.

## 2. Signal Model and Basic Principles of VAE-GAN

**2.1. DSSS Signal Model.** DSSS technology employs PN sequences independent of information data as modulation waveforms to spread signal energy over a wide bandwidth much wider than the signal information bandwidth. It is widely used in military and commercial communication networks and systems due to its antijamming, security-enhancing, and multipath fading-adaptive properties. BPSK and QPSK are the most frequently used modulation methods in the DSSS system. This paper takes the QPSK modulation signal as the sample for the experiment. QPSK-DSSS modulates the signal's homophase and orthogonal components using PN sequences of the same length and period. Figure 1 shows the structure of the QPSK-DSSS modulation.

After passing through the serial-parallel converter (S/P), the binary information bitstream enters the in-phase and quadrature branches; the information sequences entering the two branches are spread spectrum modulated using different pseudocode sequences with the same period and code chip width; the spreading signal is then carrier modulated by mutually orthogonal carriers and sent to the bandpass filter (BPF); and finally, the spreading signal is superposed to form the QPSK-DSSS signal. The QPSK-DSSS signal  $s(t)$  may be expressed as

$$s(t) = Ad_1(t)p_1(t) \cos 2\pi f_c t + Ad_2(t)p_2(t) \sin 2\pi f_c t, \quad (1)$$

where  $d_1(t)$  and  $d_2(t)$  represent two data signals in-phase and quadrature, respectively;  $p_1(t)$  and  $p_2(t)$  represent two DSSS waveforms, respectively;  $f_c$  is the carrier frequency;

and  $A$  is the energy of each binary channel symbol components.

In general, we can use the receiver to collect the radio frequency signal of the communicating party. Finally, we can obtain the in-phase/quadrature-phase (I/Q) dual-channel sampling signal at a particular intermediate frequency. For various reasons, the spreading signal collected each time may have certain differences. In order to eliminate the influence of these differences on the generative model, before training the model, it is generally necessary to perform the necessary preprocessing on the sample signal in order to increase the stability and generality of the generative model.

When preparing the sample, the collected I/Q data is first converted to zero intermediate frequency, then the signal's frequency deviation and out-of-band noise are eliminated, and the signal is resampled. The captured signal's amplitude must be normalized to accelerate the algorithm's convergence and improve the model's accuracy. To ensure that the initial phase of the spreading code is controllable in the generated associated spread spectrum waveform, we estimate the spreading code period in the DSSS signal using cepstrum technology and then divide the continuous data according to the obtained spreading code period to ensure that the training samples' initial phases remain consistent. Assuming that the linear feedback shift register of the m sequence used to generate the DSSS signal is  $n$  stages, the period length of the PN sequence is  $L = 2^n - 1$ , the oversampling multiple is  $M$ , and the length of a single sample is  $P$  spreading code periods, and then, the length of a single sample is  $L \times M \times P$ .

**2.2. Basic Principles of VAE-GAN.** GAN has been one of the fastest-growing deep learning models in recent years. It consists of a generator and a discriminator, and it learns through the two game-playing neural networks. VAE-GAN is a combination of GAN and VAE. By comparing the difference between the generated and real data, VAE will calculate the mean value to determine the loss. However, the data generated in this way is unrealistic. After adding the discriminator to the VAE, it is necessary to make the loss between the generated data and the real data smaller and let the data generated by the generator fool the discriminator so that the discriminator will force the VAE's decoder to generate more realistic data. On the other hand, GAN's generator will follow the discriminator's guidance when generating data. However, it is challenging to balance the generator's and discriminator's capabilities, which can easily result in training instability. By combining VAE and GAN, the encoder can give the generator a loss function between real and generated data so that the network is more stable while it is being trained.

Figure 2 shows the VAE-GAN model. It is mainly composed of three networks. The encoder (E) is the first component. After the encoder processes the sample  $x$ , two vectors are generated, one representing the distribution's mean vector  $\mu$  and the other representing the distribution's standard deviation vector  $\sigma$ . Then, sample  $z = E(x)$  is sampled from the sample space defined in the two vectors as the input of

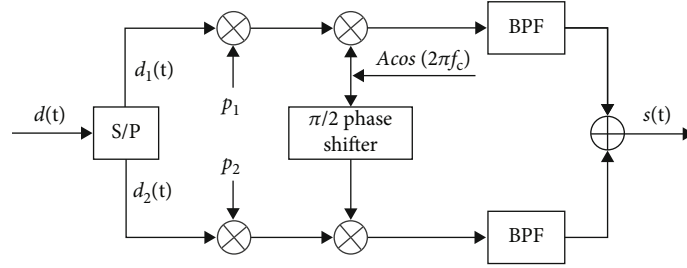


FIGURE 1: The transmitter structure of QPSK-DSSS.

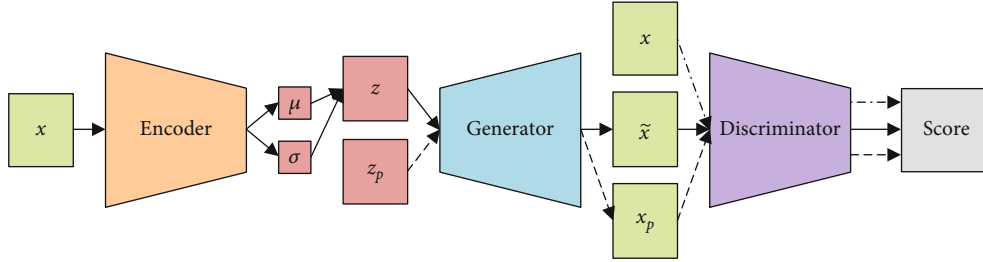


FIGURE 2: Framework of VAE-GAN.

the generator. However, the randomness of the sample will make it impossible to train these two values, so  $z$  is defined as  $z = \mu + \sigma \cdot \xi$ , where  $\xi \sim N(0, 1)$ . In this way, the randomness of the sample is transferred to  $\xi$ , so  $\mu$  and  $\sigma$  can be trained at a certain time of training. The generator (G) is the second component. The generator will obtain the sampled data  $z$  from the sample space defined by the encoder and the sampled data  $z_p$  from the normal distribution during the training process. In the generation task after the training, the generator obtains the sampled data  $z_p$  from the normal distribution. The sample space defined by the encoder must be constrained to obey the normal distribution to ensure that the generator generates meaningful data. The generator receives the sampled data  $z$  and generates the data  $\tilde{x} = G(z)$ ; similarly, the generator receives the sampled data  $z_p$  and generates the data  $x_p = G(z_p)$ . The discriminator (D) is the third component. The samples  $x$  and the generated data  $\tilde{x}$  and  $x_p$  are, respectively, input into the discriminator. The discriminator will give the sample a higher score and the generated data a lower score.

We expect that the sample space defined by the encoder is as similar to the normal distribution as possible. The distribution between the two is measured using KL divergence  $L_{\text{prior}}$ .

$$L_{\text{prior}} = D_{\text{KL}}(Q(z|x) \| P(z_p)), \quad (2)$$

where  $Q(z|x)$  represents the variational distribution encoded by the sample  $x$  and  $P(z_p)$  represents the normal distribution.

In addition, it is hoped that the greater the probability of generating real data  $x$  through the latent variable  $z$ ,

$$L_{\text{llike}}^{\text{Dis}_1} = -E_{Q(z|x)}[\log P(D(x)|z)], \quad (3)$$

where  $L_{\text{llike}}^{\text{Dis}_1}$  is to reconstruct  $x$  by maximizing the natural logarithm  $\log P(D(x)|z)$  by sampling from  $Q(z|x)$ . In specific engineering, this loss can be defined by calculating the distance between the generated data and the real data, that is, learned similarity  $\|\tilde{x} - x\|$ , so the loss function of the encoder is as follows:

$$L_E = L_{\text{llike}}^{\text{Dis}_1} + L_{\text{prior}}. \quad (4)$$

For the generator, we expect to minimize the distribution difference between the data generated by the generator using the latent variable  $z$  and the real data. Secondly, the generator also needs to receive the adversarial loss passed by the discriminator. It hopes that the data generated by the random noise and the data generated by the latent variable  $z$  can obtain higher scores in the discriminator. Therefore, the loss of the generator is as follows:

$$L_G = L_{\text{llike}}^{\text{Dis}_1} + \log(1 - D(x_p)) + \log(1 - D(\tilde{x})). \quad (5)$$

For the discriminator, the discriminator gives high scores to real samples and low scores to the data generated by the generator, so the loss of the discriminator is as follows:

$$L_D = -\log(D(x)) - \log(1 - D(\tilde{x})) - \log(1 - D(x_p)). \quad (6)$$

### 3. Methodology

**3.1. Network Architecture.** Figure 3 shows the encoder structure that we designed. Three different types of convolutional layers are used, as well as the DRSN with channel-wise thresholds (DRSN-CW) and self-attention mechanisms. In the figure, Conv1d( $k, s, p$ ) denotes a one-dimensional convolution layer in which  $k$  represents the size of the

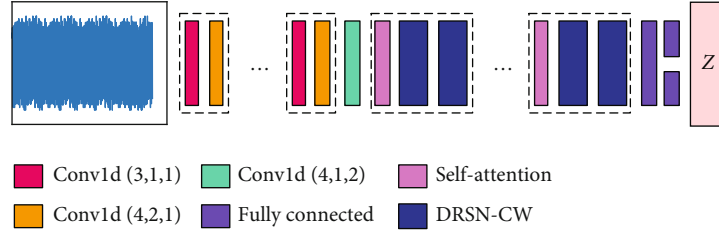


FIGURE 3: Encoder network structure.

convolution kernel,  $s$  represents the stride of the convolution, and  $p$  represents the padding added to both sides of the input. Divide the I/Q data of the sample into two channels and input them into the encoder. To begin, we use the Conv1d (3, 1, 1) and Conv1d (4, 2, 1) convolution layers to extract features in samples at varying scales. The output length remains unchanged after the data is input to the convolutional layer of Conv1d (3, 1, 1). The output length is halved after passing the data through the convolutional layer Conv1d (4, 2, 1). In Section 2.1, we mentioned that the length of a single data sample is  $L \times M \times P$ . The length of output data is reduced to  $L$  using a certain number of these two convolutional layers. The data is then supplied to the Conv1d (4, 1, 2) convolutional layer, where the length of the output data is now  $2^n$ . After the data is output from the convolution layer, it will be batch normalized, and the Leaky ReLU activation function will be used as the activation function. After the above feature extraction, we use the self-attention mechanism to more effectively consider the dependencies between the input data samples and capture the spreading sequence features. The data output from the self-attention layer does not change the length and number of channels. Next, the DRSN-CW layer is utilized to denoise the features. The purpose is to eliminate the noise in the signal as much as feasible. Too much noise makes it difficult for the network to learn the features of the original signal, and it also causes the network to learn the features introduced by the noise. The residual part in DRSN-CW is also composed of two convolutional layers, Conv1d (3, 1, 1) and Conv1d (4, 2, 1), and Figure 4 shows the DRSN-CW module.

After the data is input to the last DRSN-CW layer, the length of the output data is  $M$ , and then, the data is expanded on the full connection layer by affine transformation. The sample is converted into a 128-dimensional latent space through the reparameterization trick.

To make the trained generator generate signals faster in the actual generation task, we design the generator structure to be relatively simple and add the complexity required to complete the generation task to the encoder and discriminator. Figure 5 shows the structure of the generator. Upsample(2) indicates that upsampling multiplier is 2. The generator receives random noise as input and attempts to generate false signals with the target features. The dimension of random noise input to the generator is 128. The output length is  $256 \times L$  after the fully connected layer. Then, the data undergoes an affine transformation, the number of channels becomes 256, and the length is changed to  $L$ . After

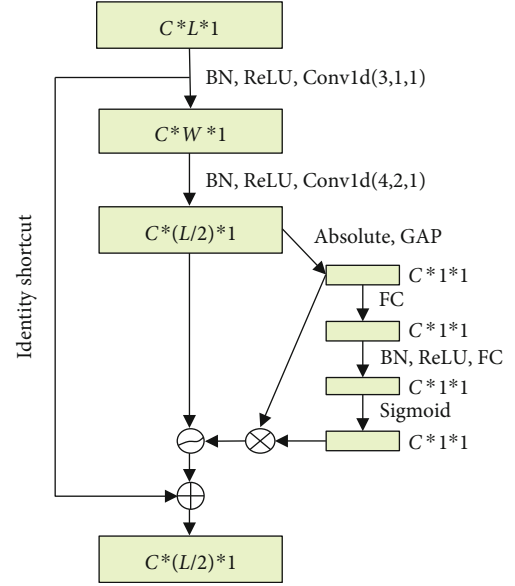


FIGURE 4: DRSN-CW module structure.

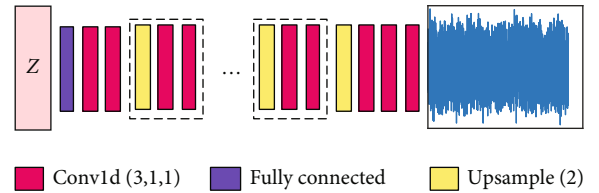


FIGURE 5: Generator network structure.

which the data passes through multiple Conv1d (3, 1, 2) layers and Upsample(2) layers, the final output channel number is 2, and the length of the generated signal is  $L \times M \times P$ .

The discriminator will distinguish whether the data is real or generated. The network structure of the discriminator we designed is similar to the encoder. Only the fully connected layer is different. Figure 6 shows the discriminator's structure. The data passes through the last DRSN-CW layer, then passes through three fully connected layers, and finally passes through the Sigmoid activation function to output the score given by the discriminator.

**3.2. Improvement of Loss Function.** Because the model's objective is to generate DSSS signals with the same characteristics as the sample, the generator should learn as much as possible about the target signal's features in order to fool

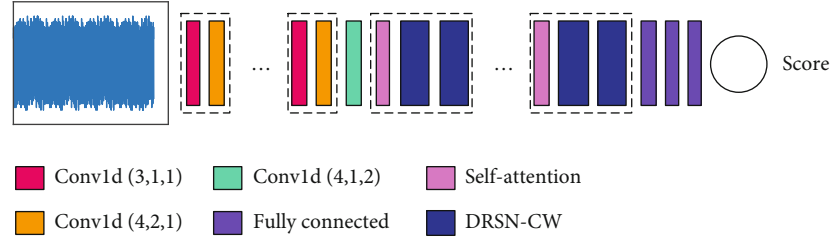


FIGURE 6: Discriminator network structure.

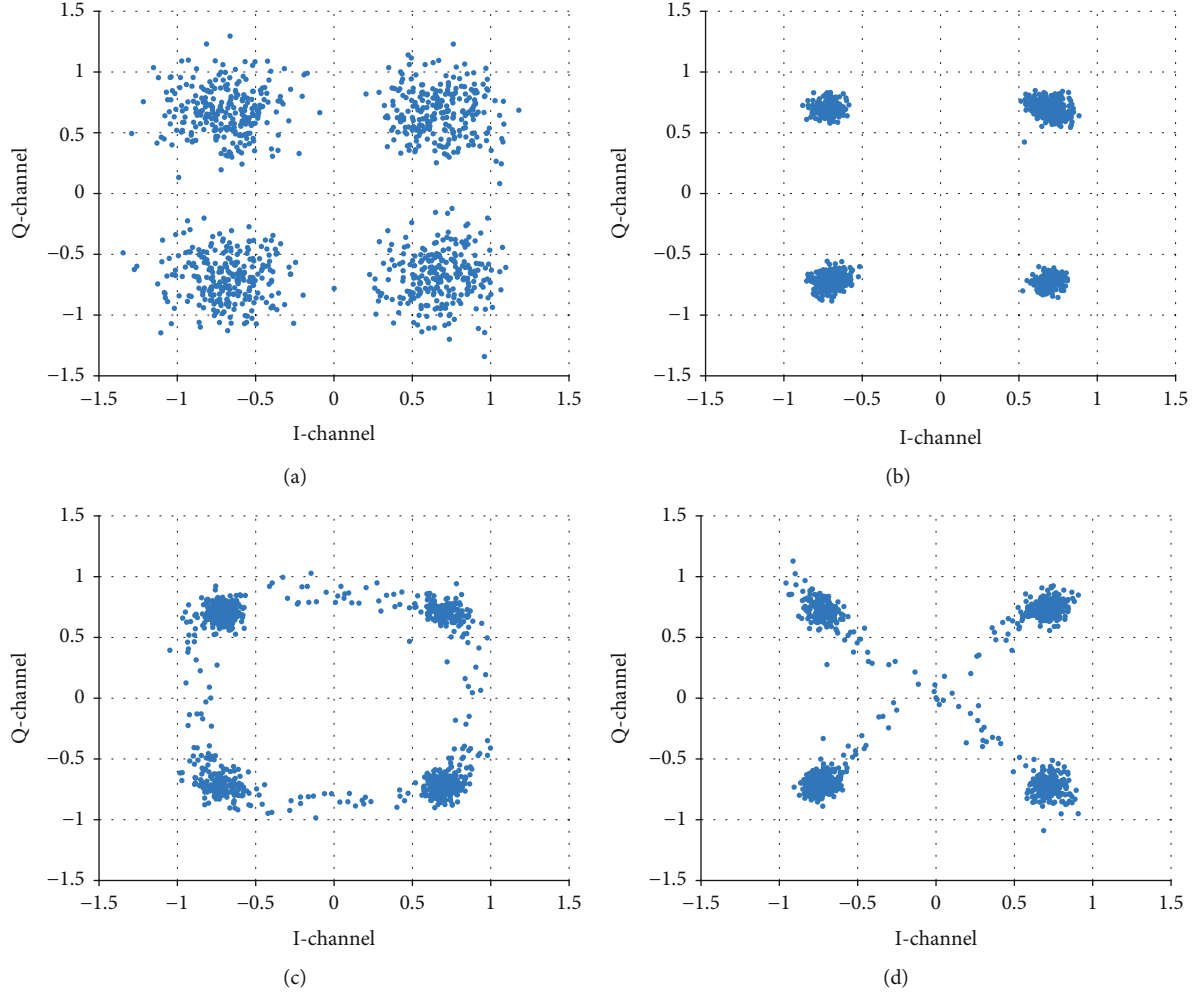


FIGURE 7: Sample and generated signal constellation.

the discriminator. However, during the early stages of GAN training, the generator’s capability is often limited. The generator cannot generate signals with a similar distribution to the sample, but the discriminator can easily identify whether the signal is real or generated. This will cause  $D(x) \rightarrow 1$  in the loss function of the discriminator, while  $D(x_p) \rightarrow 0$  and  $D(\tilde{x}) \rightarrow 0$ . When the generator is updated, the gradients of the generator are  $\partial L_G / \partial D(x_p) \rightarrow -\infty$  and  $\partial L_G / \partial D(\tilde{x}) \rightarrow$

$-\infty$ , resulting in unstable training of the generator. Additionally, the loss function of the learned similarity part in the encoder and generator cannot be used effectively, as two signals with the correct PN sequence period but opposing symbols will produce a large learned similarity loss value.

To overcome the aforementioned problems, we use the feature loss in the loss function of the generator and the encoder to replace the learned similarity loss. During

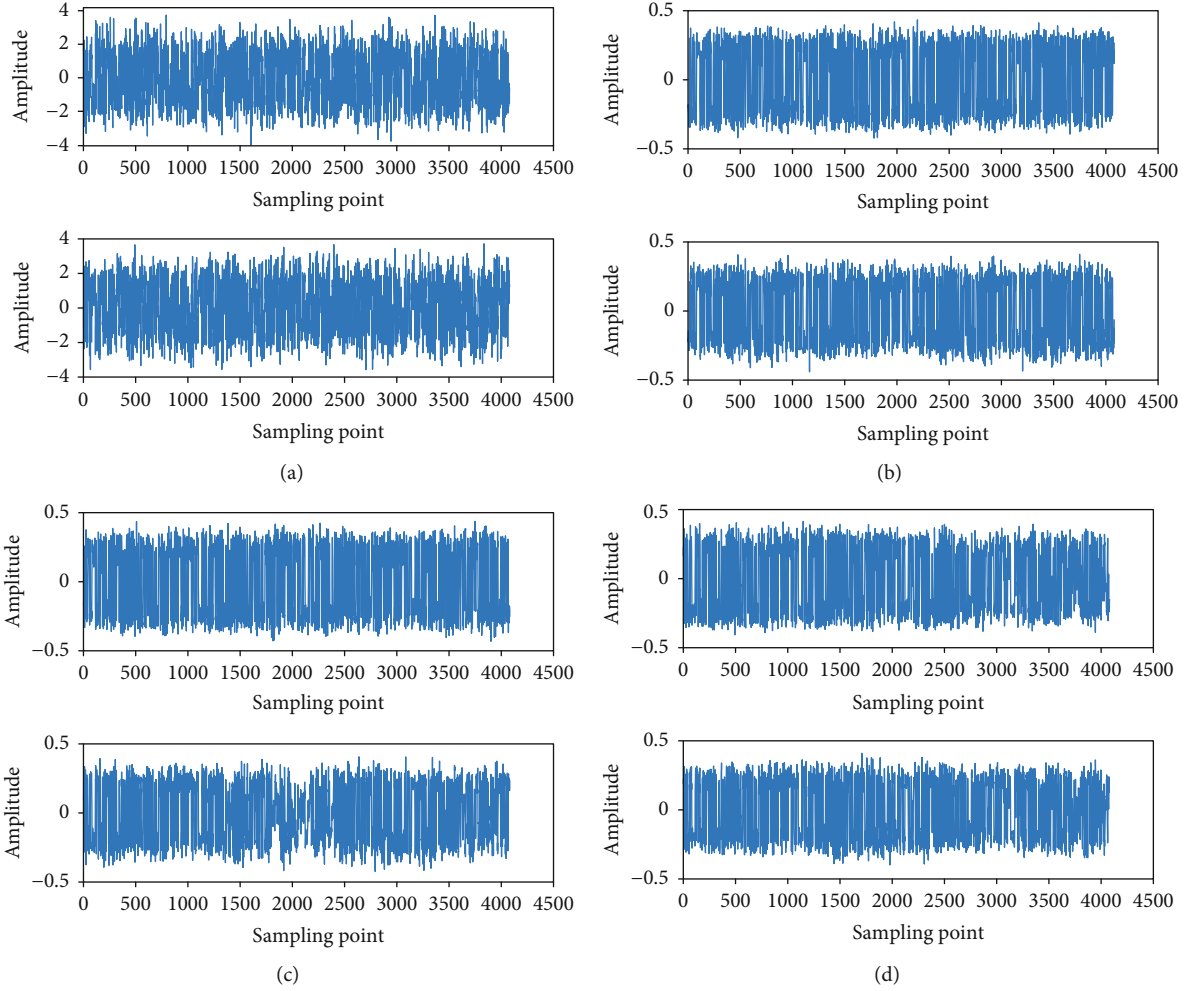


FIGURE 8: Sample and generated signal waveforms.

training, evaluate the reconstruction loss of samples and generated signals from the viewpoint of features in order to enhance the generator's and encoder's abilities. We select the output features of the fourth layer of the convolution network in the discriminator to construct the feature loss. If the fourth layer of the discriminator's feature output is  $f_D(\cdot)$ , the encoder's feature loss is

$$L_{E_{\text{feature}}} = \left\| \mathbb{E}_{x \sim p_{\text{data}}} f_D(x) - \mathbb{E}_{\tilde{x} \sim G} f_D(\tilde{x}) \right\|_2^2, \quad (7)$$

The feature loss of the generator is

$$L_{G_{\text{feature}}} = \left\| \mathbb{E}_{x \sim p_{\text{data}}} f_D(x) - \mathbb{E}_{x_p \sim G} f_D(x_p) \right\|_2^2, \quad (8)$$

So the loss function of the encoder is as follows:

$$L_{E_{\text{new}}} = L_{\text{prior}} + L_{E_{\text{feature}}} \quad (9)$$

The loss function of the generator is as follows:

$$L_{G_{\text{new}}} = \log(1 - D(x_p)) + \log(1 - D(\tilde{x})) + L_{G_{\text{feature}}}. \quad (10)$$

We still use the discriminator loss from the original VAE-GAN.

## 4. Experimental Results and Analysis

**4.1. Experimental Environment.** The simulation experiments were performed on an Intel Core i9-10980XE desktop computer with a 3.00 GHz CPU and 256 GB of RAM, using two NVIDIA Quadro GV100 graphics cards. The Python version is 3.8. The PyTorch version is 1.8.

**4.2. Experimental Data Set.** The experiment uses four types of samples with PN sequence period lengths of 31, 63, 127, and 255. The signal-to-noise ratio (SNR) is 5-15 dB, the noise is additive white Gaussian noise (AWGN), and the number of samples of each type is 5000. Each sample contains four symbol lengths, and the start position of the sample signal data is not the start position of the PN sequence period.

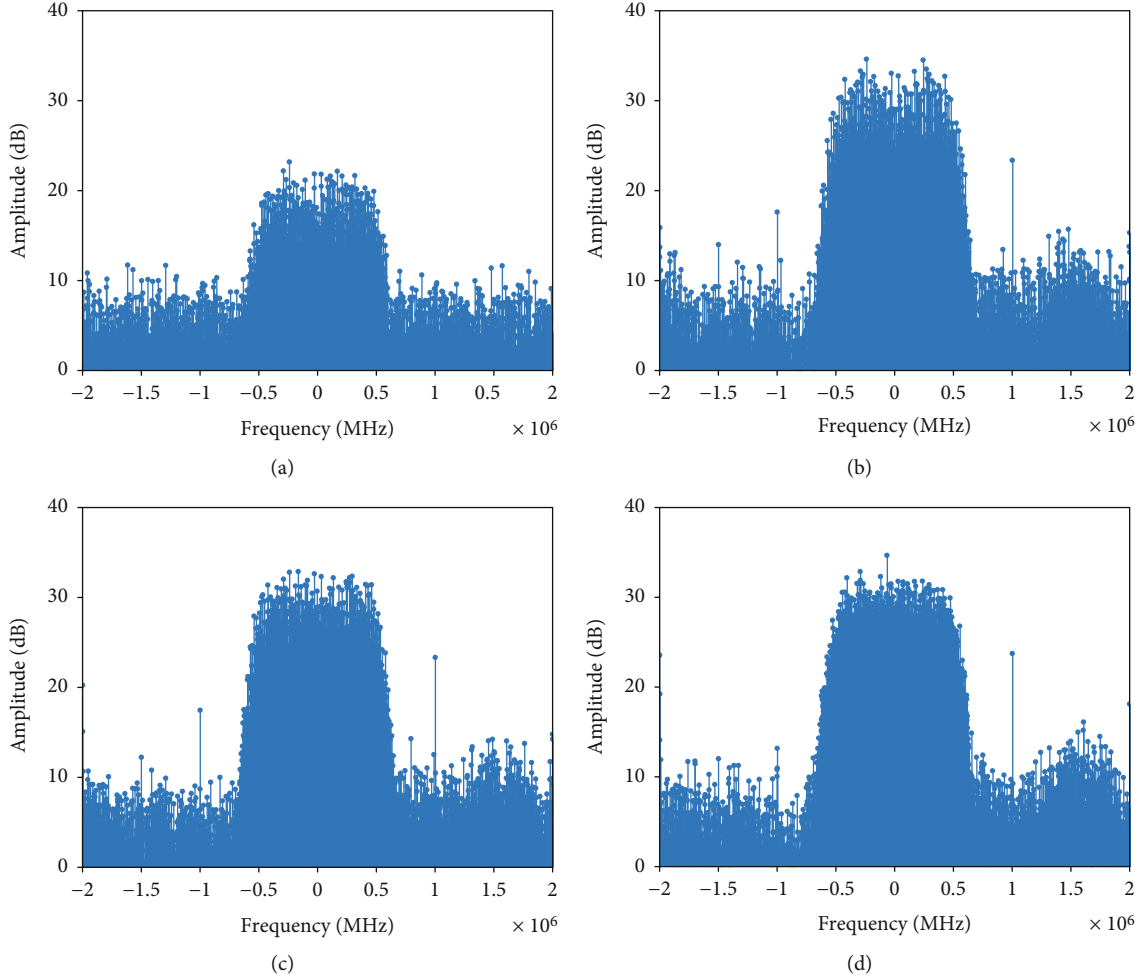


FIGURE 9: Sample and generated signal power spectral density.

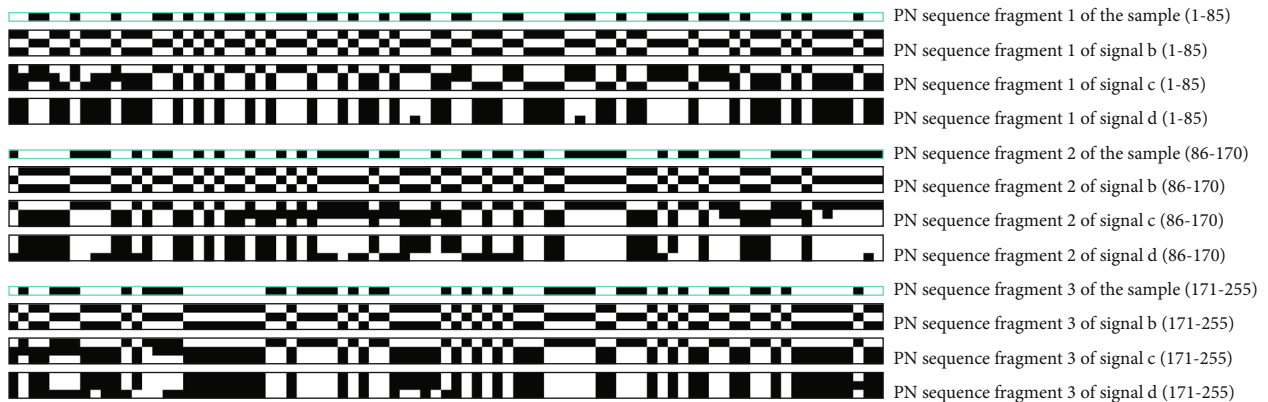


FIGURE 10: PN sequence of samples and generated signals.

We train the network for 300 epochs using Adam with a batch size of 16. The coefficients used for computing running averages of gradient and its square are 0.5 and 0.9, respectively. The learning rate of the encoder is set to 0.0001, the learning rate of the generator is set to 0.0003, and the learning rate of the discriminator is set to 0.0001.

**4.3. Experimental Results and Analysis.** Samples are fed into the network with experimental parameter set to train the model, using the VAE-GAN network architecture proposed in this paper. After the model training is completed, the samples are sampled from the random noise that obeys the normal distribution and input to the generator for signal

generation. Analyze the quality of the generated signal by comparing it with the sample.

Assuming that the length of the spreading code is 255, used by the sample, and the SNR is 10 dB, the generated results are displayed and analyzed. By inputting random noise into the generator to generate 1000 signals, the constellation diagrams of the generated signals can be divided into three categories, which are represented by the constellation diagrams shown in Figures 7(b)–7(d). The signals represented by these three constellation diagrams are hereinafter referred to as signals b, c, and d, respectively, where a is a sample constellation diagram. Figure 8 shows a time-domain waveform diagram corresponding to the constellation diagram in Figure 7. It can be seen that the constellation diagram of signal b conforms to QPSK modulation, and the constellation points are relatively dense. In the time-domain waveform, the amplitude of the waveform diagram of signal b is more regular than that of the sample signal. The constellation diagram of signal c appears as a ring. From the waveform diagram, it can be observed that the real part of the waveform of signal c is normal in the time domain, and the amplitude of the imaginary part waveform varies greatly at different sampling points, resulting in a ring on the constellation diagram. The constellation diagram of signal d depicts an “X” shape. From the waveform diagram, it can be seen that the amplitudes of the real part and imaginary part of the signal are lower at the sampling points at some corresponding positions. Without regard to the correctness of the spreading code in the signal, it can be considered that signal b is of high quality. In contrast, signals c and d reflect some distortion.

Estimate the power spectral density of the sample and the generated signal. The power spectral density of the sample is shown in Figure 9(a), and the power spectral density of the generated signal is shown in Figures 9(b)–9(d). It can be seen from Figure 9 that the spectrum of the generated data is located near the zero frequency, the generated signal is similar to the sample bandwidth, and the SNR of the generated signal is higher than that of the sample. Furthermore, we observed the difference in SNR between samples and generated signals at different SNRs and found no strict correspondence between them. In general, the SNR of the generated signal is 0-10 dB higher than the SNR of the samples.

Estimate the spreading code on each information bit of the generated signal. Count the number of chips that are consistent between the generated spreading code and the sample spreading code, and calculate the ratio of this value to the code length. Because the data generated by GAN is random, it is impossible to determine whether the spreading code of a certain information bit of the generated signal is the PN code of the sample or the inverse of the PN code, and then, there will be two different ratios. We take the higher ratio as the accuracy of the final generated spreading code and use its corresponding code to determine the data information bits. The closer the accuracy rate is to 50%, the worse the effect of the generated spreading code. We expect that the model can generate the spreading code with an accuracy rate of over 85%.

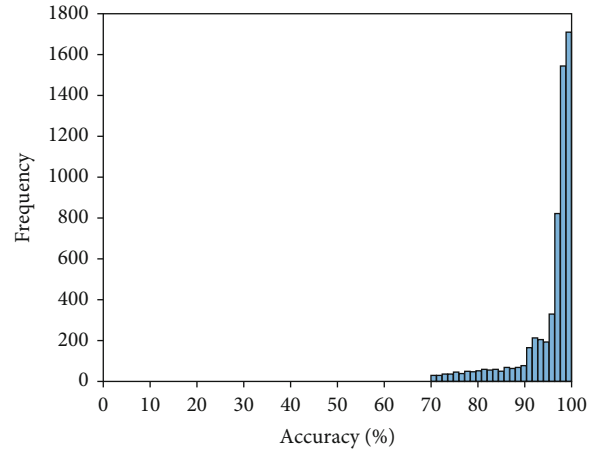


FIGURE 11: The histogram of the accuracy distribution of the PN sequence of the generated signal.

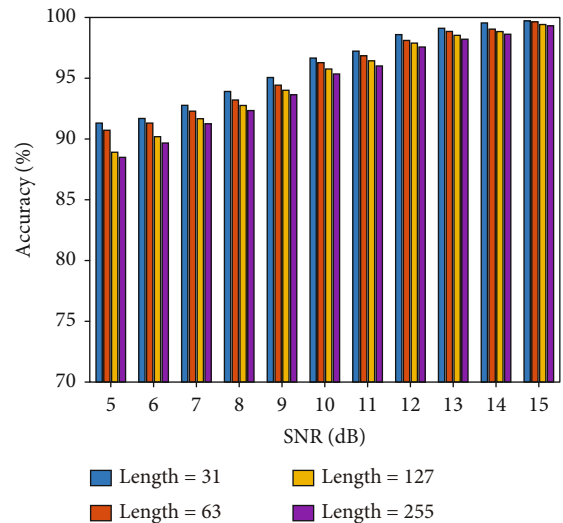


FIGURE 12: Average accuracy of PN sequence in generated signal.

The sample contains four spreading code period lengths. In this experiment, the starting position of the sample is different from the starting position of the spreading code. Experiments show that the generated signal and sample have consistent starting positions in the spreading code periods and that the generated signal’s I/Q contains three complete spreading code periods. To show the spreading code generated by signals b, c, and d, we extract the spreading code generated by the sample’s I channel, as illustrated in Figure 10. The white block represents chip “1,” and the black block represents chip “0.” Each complete sequence period is in a row, and each generated signal of the I channel has three rows. Due to space constraints, we have divided the entire period of the spreading code into three fragments for presentation. It can be concluded that the accuracy of the three spreading codes generated by signal b is 100%; the accuracy of the three spreading codes generated by signal c is 99.61%, 77.65%, and 100%, respectively; and the accuracy of the three spreading codes generated by signal d is 100%,



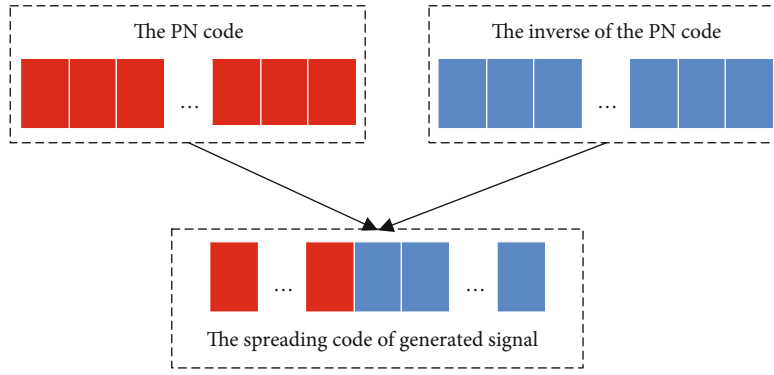


FIGURE 13: Partial spreading codes generated without self-attention mechanism.

99.61%, and 100%, respectively. Additionally, the three data signals generated contain different information.

Calculate the accuracy of the spreading code of 1000 generated signals. Each generated signal contains six complete spreading codes, resulting in a total of 6000 complete spreading codes. Calculate the accuracy of each spreading code separately. Figure 11 shows the spectrum histogram. When the spreading code length of the sample is 255, and the SNR is 10 dB, the accuracy rate of the PN sequence of the generated signal is greater than 70%, and the majority of them are between 90% and 100%. The average accuracy is 95.35%.

For the four types of samples with spreading code lengths of 31, 63, 127, and 255, we generate 1000 signals, respectively, when the SNR is 5-15 dB, and calculate the average accuracy rate of the spreading code of each generated signal. Figure 12 shows the results. At the same length of spreading code, with the improvement of the sample's SNR, the accuracy of the PN sequence of the generated signal is also gradually improved. The accuracy of the PN sequence of the generated signal is gradually reduced with an increase in the length of the spreading code of the sample.

We evaluated the model's performance without applying the self-attention mechanism. When the length of the spreading code used by the sample is 63 or less, there is little difference in the accuracy of the PN sequence of the generated signal whether the self-attention mechanism is used or not. When the length of the spreading code used by the sample is more than 63, the spreading code of some information bits of the signal generated by the network that does not use the self-attention mechanism will produce the phenomenon shown in Figure 13. In the figure, the red rectangle represents the PN sequence of the sample, and the blue rectangle represents its inverse PN sequence. The spreading code of a certain information bit in the generated signal will be spliced from the PN code and the inverse PN code. After testing 1000 generated signals, this situation usually occurs in one information bit of a generated signal, while the spreading code of other information bits is standard. This kind of signal accounts for about 18% of the total signal generated. However, if the self-attention mechanism is used, this situation will not occur.

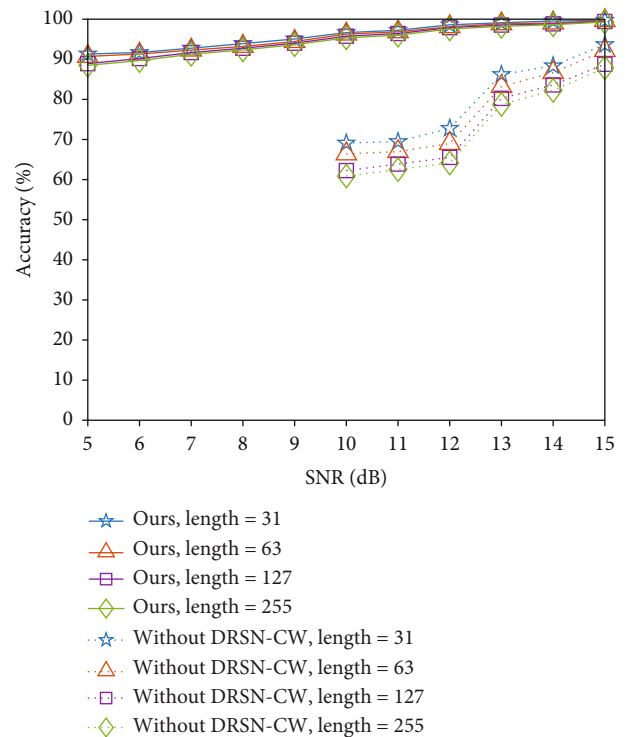


FIGURE 14: Comparison of average accuracy between the proposed model and the model without DRSN-CW.

We also tested the performance of the model without DRSN-CW. Figure 14 shows the performance comparison between the model and the model in this paper under different length spreading codes and different SNRs. Under the same code length and SNR, the average accuracy of the spreading codes generated by the model is far lower than the model in this paper, and they are sensitive to the changes in the SNR of samples. When the SNR is lower than 13 dB, the model's performance drops sharply. It can be seen that the spreading code of the generated signal at 10-12 dB is far below our tolerable error range, so we have not simulated the SNR below 10 dB. In this case, the network has been unable to effectively learn the PN sequence characteristics of the sample.

## 5. Conclusion

In this paper, we designed a GAN capable of reconstructing DSSS signals. This method avoids complex parametric analysis of the signal. The generation process is simple, and the model can be adapted to generate DSSS signals with varying PN sequence period lengths. This method is of great significance to the construction of a complex electromagnetic environment. In order to make the network learn the PN sequence characteristics of the DSSS signal, we introduced DRNs and convolution modules based on the self-attention mechanism into the encoder and discriminator in VAE-GAN. The DRN layer will denoise the sample. The self-attention network layer can capture the features of the PN sequence of the sample. Through the adversarial training in VAE-GAN, the generated signal with the characteristics of the target signal is finally obtained. The results show that the signal generated by the proposed network can learn the modulation pattern and spectral features of the sample, as well as the sample's PN code sequence. Although the network proposed in this paper is suitable for PN sequences with different code lengths, the effect in low SNR is unsatisfactory. The next step is to study the generation of DSSS signals in the lower SNR ratio. In addition, the I/Q in this paper uses the same PN sequence for modulation. Next, we will try to use different PN sequences.

## Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## References

- [1] J. Y. Zhang and G. H. Wei, "New Development of Electromagnetic Compatibility in the Future: Cognitive Electromagnetic Environment Adaptation," in *2021 13th Global Symposium on Millimeter-Waves & Terahertz (GSMM)* pp. 1–3, China, May, Nanjing, China, 2021.
- [2] V. G. Maksimenko, "Compensation for man-made interference during reception of an ultra-low-frequency electromagnetic field in the sea," *Journal of Communications Technology and Electronics*, vol. 65, no. 2, pp. 127–130, 2020.
- [3] P. Händel and D. Rönnow, "MIMO and massive MIMO transmitter crosstalk," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1882–1893, 2019.
- [4] M. R. Ahmad, M. R. M. Esa, V. Cooray, and E. Dutkiewicz, "Interference from cloud-to-ground and cloud flashes in wireless communication system," *Electric Power Systems Research*, vol. 113, pp. 237–246, 2014.
- [5] M. Q. Liu, Z. L. Liu, W. D. Lu, Y. Chen, X. Gao, and N. Zhao, "Distributed few-shot learning for intelligent recognition of communication jamming," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 395–405, 2022.
- [6] M. Q. Liu, J. K. Wang, N. Zhao, Y. Chen, H. Song, and R. Yu, "Radio frequency fingerprint collaborative intelligent identification using incremental learning," *IEEE Transactions on Network Science and Engineering*, p. 1, 2021.
- [7] Y. G. Tang, T. T. Zhang, and X. J. Zhou, "Construction and Evaluation of Examination Conditions for Complex Electromagnetic Environment Based on Subject and Object," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)* pp. 1174–1180, China, May, Chongqing, China, 2019.
- [8] P. Zhou, "Overview on communication countermeasure jamming effect assessment method," *Communications Technology*, vol. 49, no. 8, pp. 1029–1033, 2016.
- [9] X. W. Gu, Z. J. Zhao, and S. Lei, "Blind estimation of pseudo-random codes in periodic long code direct sequence spread spectrum signals," *IET Communications*, vol. 10, no. 11, pp. 1273–1281, 2016.
- [10] J. B. Long, H. B. Wang, D. F. Zha, H. Fan, Z. Lao, and H. Wu, "Applications of an improved time-frequency filtering algorithm to signal reconstruction," *Mathematical Problems in Engineering*, vol. 2017, 14 pages, 2017.
- [11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.
- [12] J. Qin, *Signal reconstruction based on generative adversarial networks*, Xidian University, 2018.
- [13] H. J. Yang, L. Chen, and J. Y. Zhang, "Research on digital signal generation technology based on generative adversarial network," *Electronic Measurement Technology*, vol. 43, no. 20, pp. 127–132, 2020.
- [14] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning - WiseML 2019*, pp. 55–60, Miami FL USA, 2019.
- [15] F. Zhao and H. Jin, "Communication jamming waveform generation technology based on GAN," *Systems Engineering and Electronics*, vol. 43, no. 4, pp. 1080–1088, 2021.
- [16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International Conference on Machine Learning*, pp. 1558–1566, New York NY USA, 2016.
- [17] M. H. Zhao, S. S. Zhong, X. Y. Fu, B. P. Tang, and M. Pecht, "Deep residual shrinkage networks for fault diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4681–4690, 2019.
- [18] W. B. Jiang and A. Liu, "Image motion deblurring based on deep residual shrinkage and generative adversarial networks," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 5605846, 15 pages, 2022.
- [19] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.