HpGAN: Sequence Search With Generative Adversarial Networks

Mingxing Zhang, Zhengchun Zhou, Lanping Li, Zilong Liu, Meng Yang, and Yanghe Feng

Abstract— **Sequences play an important role in many engineering applications. Searching sequences with desired properties has long been an intriguing but also challenging research topic. This article proposes a novel method, called HpGAN, to search desired sequences algorithmically using generative adversarial networks (GANs). HpGAN is based on the idea of zero-sum game to train a generative model, which can generate sequences with characteristics similar to the training sequences. In HpGAN, we design the Hopfield network as an encoder to avoid the limitations of GAN in generating discrete data. Compared with traditional sequence construction by algebraic tools, HpGAN is particularly suitable for complex problems which are intractable by mathematical analysis. We demonstrate the search capabilities of HpGAN in two applications: 1) HpGAN successfully found many different mutually orthogonal complementary sequence sets (MOCSSs) and optimal odd-length binary Z-complementary pairs (OB-ZCPs) which are not part of the training set. In the literature, both MOCSSs and OB-ZCPs have found wide applications in wireless communications and 2) HpGAN found new sequences which achieve a four-times increase of signal-to-interference ratio—benchmarked against the well-known Legendre sequences—of a mismatched filter (MMF) estimator in pulse compression radar systems. These sequences outperform those found by AlphaSeq.**

Index Terms— **Generative adversarial networks (GANs), Hopfield network, mutually orthogonal complementary sequence set (MOCSS), odd-length binary Z-complementary pairs (OB-ZCPs), pulse compression radar.**

I. Introduction

A SEQUENCE is a list of elements arranged in a specific order. Good sequences form a core component of many information systems. As rapidly evolving wireless mobile communication technologies meet the increasingly stringent and diverse requirements of various data services, it is critical to design sequences with different properties. For example, sequences with low autocorrelation are widely used in pulse compression radars, sonars and channel synchronization of digital communication [1]. Also, orthogonal sequence sets are used to distinguish signals from different users in cellular code-division multiple access (CDMA) systems [2].

In the open literature, sequences are commonly designed by algebraists using mathematical tools such as finite field theory, algebraic number theory, and character theory. However, in practical scenarios, it may be difficult to construct sequences using specific mathematical tools. To facilitate the use of mathematical tools, some constraints in terms of sequence lengths, alphabet size, and set size may be imposed. For example, Davis and Jedwab constructed polyphase Golay complementary sets of sequences by generalized Boolean function, but the sequence lengths are restricted to 2^m, where m is nonnegative integer numbers [3].

Algorithm design is another direction of obtaining good sequences. A key issue here is whether good sequences can be found within a reasonable time by algorithms. There are two major types of sequence search algorithms: one is optimization algorithms through specific mathematical analysis, and the other is heuristic algorithms that generate high-quality solutions in a reasonable time for practical use, although there is no guarantee of finding a globally optimal solution [4]. A good optimization algorithm can effectively find sequences with guaranteed convergence [5]–[8]. However, such optimization algorithms need to be carefully designed case by case and their derivations may not be straightforward. Heuristic algorithms are not so sensitive to specific problems and can handle complex optimization problems efficiently, such as simulated annealing [9], [10], evolutionary algorithm [11], [12], and neural network [13].

In principle, deep learning is based on an architecture of linearly weighted processing layers that are connected mimicking the neurons in a human brain. In recent years, deep learning has provided a significant improvement in performance compared with conventional hand-crafted schemes in many fields [14], [15]. Far more than merely a tool for cognitive tasks such as image recognition, deep learning shows great potential for applications in communication systems. For instance, the end-to-end optimization of the physical layer of a communication system by autoencoder has led to enhanced performances than conventional solutions [16]–[21].

Manuscript received December 9, 2020; revised June 19, 2021 and September 5, 2021; accepted November 1, 2021. The work of Mingxing Zhang and Zhengchun Zhou was supported in part by the National Natural Science Foundation of China under Grant 62071397 and Grant 62131016. *(Corresponding authors: Zhengchun Zhou; Yanghe Feng.)*

Mingxing Zhang and Zhengchun Zhou are with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail: mingxingzhang@my.swjtu.edu.cn; zzc@swjtu.edu.cn).

Lanping Li is with the School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China (e-mail: lilanping523@gmail.com).

Zilong Liu is with the School of Computer Science and Electronics Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: zilong.liu@essex.ac.uk).

Meng Yang is with the School of Mathematics, Southwest Jiaotong University, Chengdu 610031, China (e-mail: mekryang@gmail.com).

Yanghe Feng is with the College of Systems Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: fengyanghe@nudt.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3126944.

Digital Object Identifier 10.1109/TNNLS.2021.3126944

In [17] and [18], the constellation mapping and demapping of symbols on each subcarrier are determined adaptively through an autoencoder in order to jointly optimize both the bit error rate (BER) and the peak-to-mean envelope power ratio (PMEPR) of an orthogonal frequency division multiplexing (OFDM) system. In the field of communication security, to prevent external intrusions, clustering algorithms have been combined with neural networks to enhance the system's ability to detect abnormal data [19], [22], [23]. Moreover, deep learning and reinforcement learning (RL) have been employed to improve the spectrum allocation so as to avoid mutual interference [24]–[29].

Deep learning has also received considerable attention in the field of sequence design owing to its capability of approximating complex nonlinear functions and various optimization methods [30], [31]. In 2020, based on the basic framework of AlphaGo, Shao *et al.* [30] proposed a new RL model AlphaSeq to discover sequences. AlphaSeq treats the sequence discovery problem as an episodic symbol-filling game, in which a player fills symbols in the vacant positions of a sequence set sequentially during an episode of the game. In [31], the deep residual neural network constructed with the sequence metric as the loss function showed better results than the existing optimization algorithms. However, the above two methods are limited in some special scenarios in which the design criteria for good sequences are complex or may not be formulated by a mathematical expression. In this work, we focus on a novel network structure based on generative adversarial networks (GANs) to tackle the aforementioned problem.

GAN is a class of generative models that have been proposed by Goodfellow *et al.* [32]. GAN consists of a generative net G and discriminative net D, the framework of which corresponds to a minimax two-player game. Specifically, the generative net G is used to capture the data distribution, whilst the discriminative net D helps estimate the probability that a sample came from the training data rather than G. This approach has been widely applied in computer vision for generating samples of natural images [33].

That being said, GAN is designed to generate real-valued continuous data and may have some limitations when dealing with discrete data. The reason is that the discrete output of the generative model makes it difficult to transfer the gradient update from the discriminant model to the generative model [34], [35]. Yu *et al.* [35] proposed a generative model SeqGAN that can generate discrete data. The SeqGAN directly performs gradient policy updates to bypass the generator differentiation problem. This is done by combining RL and GAN, using the output of D as a reward for RL, and then updating G with the policy gradient of RL. SeqGAN has achieved remarkable successes in natural language (such as speech language and music generation) by using recurrent neural networks (RNNs) as the generator G. Similar to AlphaSeq, SeqGAN generates a sequence by symbol filling, where a symbol is generated each time with RNN guessing. However, the discriminator D can only assess the reward of the complete sequence, and cannot evaluate the reward of the current generated partial sequence and its impact on the subsequent generated complete sequence. Thus, to evaluate the score for an intermediate symbol, the authors applied a Monte Carlo search to sample the unknown last remaining symbols. As a result, a high computational complexity may be inevitable.

In this article, aiming for sequences that may be difficult to generate with systematic constructions, we propose a new network architecture called HpGAN. In HpGAN, we design the encoder and decoder through the Hopfield network [36], [37], where the encoder can convert discrete data into continuous data, and the decoder can restore continuous data to discrete data. The encoding module has two main functions: one is to enable GAN to produce continuous data and the other is to solve the problem of small samples faced by GAN. Both the generative net G and the discriminative net D in HpGAN are multiple layers of perceptron (MLP), which are easy to implement with lower computational complexity than other networks, such as convolutional neural networks (CNNs).

The major contributions of this work are summarized as follows.

1) We have proposed a new GAN-based paradigm, HpGAN, to discover binary sequences with the desired correlation properties. In HpGAN, the Hopfield network is adopted to design decoder and encoder, which solves the problem of GAN in generating discrete sequences. Since the encoder and decoder only require a few simple vector operations, they can be readily applied to existing frameworks with marginal training overheads.
2) We have employed HpGAN to search for the following two classes of sequences that are not from the training set.
    a) Complementary and near-complementary sequences that have found wide applications in wireless communications such as interference suppression and PMEPR reduction in multicarrier CDMA (MC-CDMA) systems.
    b) Phase-coded sequences for pulse compression radar systems. The generated sequences display a significant improvement in performance when compared with the well-known Legendre sequences and the sequences generated by AlphaSeq.

The remainder of this article is organized as follows. Section II formulates the sequence search problem and outlines the framework of HpGAN. Sections III and IV present the applications of HpGAN for Optimal Sequence Sets and phase-coded sequences, respectively. Section V concludes this article. Throughout this article, lowercase bold letters denote vectors and uppercase bold letters denote matrices.

II. Methodology

A. Problem Formulation

Let $\mathcal{C} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ denote a binary sequence set which consists of M different sequences of the same length N, where the kth sequence is given by $\boldsymbol{x}_k = [x_1, x_2, \ldots, x_N]^T$ with $x_i = \{-1, 1\}$ for all $1 \leq i \leq N$. Let $\mathcal{M}(\mathcal{C})$ be a measure of the goodness of sequence set $\mathcal{C}$. Our objective is to find a sequence set $\mathcal{C}^*$ with the best $\mathcal{M}^*$ with certain criteria. For example, an optimal sequence set whose maximum cross correlation meets the celebrated Welch bound is desired for the mitigation of multiuser interference when it is applied in CDMA. Exhaustive search of optimal binary sequence sets may be infeasible due to the prohibitively high complexity of $O(2^{MN})$.

Let $P_{\mathcal{C}}$ be the probability distribution function of all sequence sets with $\mathcal{M}(\mathcal{C}) \leq \xi$, i.e., $P_{\mathcal{C}} = P(\mathcal{M}(\mathcal{C}) \leq \xi)$, where $\mathcal{C} \in \Omega$ and Ω denote the solution space of all binary sequence sets $\mathcal{C}$. The problem of sequence search can be transformed into the task of training a generative model G to learn the probability distribution function $P_{\mathcal{C}}$. By doing so, new sequence sets can be obtained with probability distribution function P_G. We adopt the idea of GAN by regarding the task of learning the probability distribution function $P_{\mathcal{C}}$ as a zero-sum game. The core idea of GAN is based on Nash equilibrium in game theory. It sets the two parties involved in the game as a generator G and a discriminator D. The purpose of the G is to learn the real data distribution $P_{\mathcal{C}}$ as much as possible, while the purpose of the D is to judge whether the input data comes from $P_{\mathcal{C}}$ or P_G as much as possible. In order to win the game, these two players need to constantly optimize and improve their generating ability and discriminating ability, respectively, until a Nash equilibrium is attained.

In general, the calculation of the metric function $\mathcal{M}(\mathcal{C})$ is time-consuming. For some special scenarios, it may not be possible to use mathematical formulas to accurately measure the properties of sequences. Fortunately, GAN avoids the above problems by learning the probability distribution $P_{\mathcal{C}}$ of the training sequences. In GAN, D and G play the following two-player minimax game with value function $V(G, D)$ [32]:

$$\min_G \max_D V(G, D) = E_{\mathcal{C} \sim P_{\mathcal{C}}(\mathcal{C})}\big[\log D(\mathcal{C})\big] + E_{z \sim P_z(z)}\big[\log(1 - D(G(z)))\big]$$

where z and P_z represent noise and noise probability distributions, respectively.

B. Methodology

It is necessary to obtain a large amount of training data when searching sequences by GAN. It is easy to generate sequences with good rather than the best $\mathcal{M}(\mathcal{C})$ as training data by existing algorithms or constructions, such as stochastic search methods. Unfortunately, the application of GAN suffers from two problems. First, GAN is designed for generating real-valued, continuous data but has difficulties in directly generating sequences of discrete data, such as discrete phase sequences. This is because in GANs, the generator starts with random sampling, followed by a determistic transform, according to the model parameters. As such, the gradient of the loss from discriminant model D with respect to the outputs by G is used to guide the generative model G (parameters) to slightly change the generated value to make it more realistic. If the generated data are based on discrete phases, the "slight change" guidance from the discriminative net makes little sense because there may be no corresponding phase for such slight change in the limited dictionary space. Second, it is difficult for GAN to learn the real distribution, when there is a small amount of training data. Roughly speaking, the more sampled data we have, the closer the learned probability distribution approaching the true one. However, even with the existing algorithms or constructions, the available training datasets are still very limited.

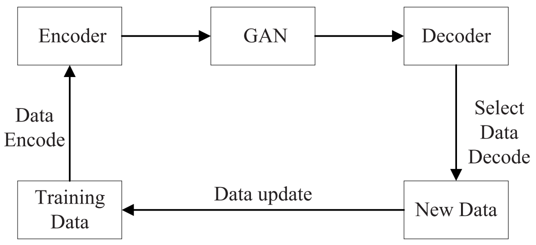


Fig. 1. Iterative algorithmic framework of HpGAN. The training dataset is iteratively updated to improve the search capability of GAN until equilibrium is attained.

In this article, to address the above two difficulties, we develop a new network framework called HpGAN by combining Hopfield network and GAN, in which we design an encode method to map binary sequences to a continuous interval. Then we exploit the discrete Hopfield network framework to decode the generated continuous datasets into the binary ones when GAN reaches equilibrium. The overall algorithmic framework of HpGAN is as shown in Fig. 1. First of all, we generate a dataset through the existing algorithm or structure as the initial sample dataset, compile the discrete dataset into a continuous dataset by using the encoder and then feed the encoded dataset into GAN. A generator G which can generate data similar to the training data is obtained when the GAN reaches the Nash equilibrium. Second, the generator obtains a large amount of data and selects some good datasets as new samples to update the initial samples. Through the iterative process, GAN can progressively generate a better dataset than the initial samples.

In what follows, we introduce these components and the relationship between them in more detail.

1) Encoder: The encoder is designed not only to transform discrete data into continuous data but also plays an important role in expanding the training dataset. The encoder is defined as follows.

Definition 1: Let $\mathbf{S}_P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_P\}$ be a sample dataset with P discrete sequences, then the corresponding continuous sample is

$$\boldsymbol{y} = \frac{1}{N}\left[\sum_{p=1}^{P}\Big[\boldsymbol{x}_p(\boldsymbol{x}_p)^T - \boldsymbol{I}\Big] - \mathbf{b}\right] \tag{1}$$

where N is the length of sequence x_p, $\boldsymbol{I}$, and $\boldsymbol{b}$ are the identity matrix and bias vector, respectively.

In HpGAN, in order to eliminate the dimensional influence between data features, we limit the data normalization to $[-1, 1]$, i.e., $(1/N)\sum_{i=1}^{P}[\boldsymbol{x}_p(\boldsymbol{x}_p)^T - \boldsymbol{I}]$. The bias vector $\boldsymbol{b}$ represents random noise, which is used to increase the tolerance of encoding. Another advantage is to improve the exploration ability of the model. The specific steps of the encoding process are as follows.

1) Choose a number P less than $\mathcal{N}$ at random, i.e., $P \leq \mathcal{N}$, where $\mathcal{N}$ is *a priori* constant.

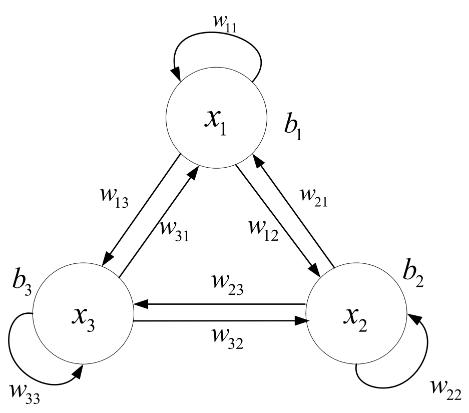


Fig. 2. DHNN architecture with three neurons, in which neuron i is connected to neuron j with the weight $w_{ij} \in (-1, 1)$, and b_i is the bias of neuron i. Each neuron is connected to all neurons of the network.

2) Choose P samples in $\mathcal{S}$ at random, and transform them into continuous sample by (1).
3) Repeat the above steps until enough datasets are generated.

In this way, we can obtain sufficient different datasets to effectively solve the problem of small samples. In Section II-B.2, we will explain the validity of continuous datasets.

2) Decoder: In this section, we introduce our designed decoder based on the discrete Hopfield neural network (DHNN) framework. DHNN is a single-layer full feedback neural network with n neurons. Fig. 2 shows a DHNN architecture with three neurons, where w_{ij} denotes the weight value connecting neuron i to neuron j and b_i is the bias of neuron i. x_i is called the state of neuron i, and the set $\boldsymbol{x} = \{x_1, x_2, x_3\}$ of all neuron states constitutes the state of feedback network. The input of the feedback network is the initial value of the network, which is expressed as $\boldsymbol{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\}$. In the dynamic evolution process of the network from the initial state $\boldsymbol{x}^{(0)}$, the state transition rule of each neuron can be expressed as follows:

$$x_j = f\big(\mathrm{net}_j\big), \quad j = 1, 2, 3 \tag{2}$$

where $f(\cdot)$ denotes the activate function. Specifically, the symbolic function is used as $f(\cdot)$ in DHNN, that is,

$$f\big(\mathrm{net}_j\big) = \mathrm{sgn}\big(\mathrm{net}_j\big) = \begin{cases} 1, & \mathrm{net}_j \geq 0 \\ -1, & \mathrm{net}_j < 0. \end{cases} \tag{3}$$

The net_i is the net input of neuron i, which be defined as

$$\mathrm{net}_j = \sum_{j=1}^{n} \big(w_{ij} x_i - b_i\big), \quad i = 1, 2, 3 \tag{4}$$

where $w_{ii} = 0, w_{ij} = w_{ji}$.

In DHNN, there are two dynamic evolution methods: synchronous update and asynchronous update. In the synchronous update mode, all neurons in the network adjust the state at the same time. In contrast, in the asynchronous update mode, only one neuron state is adjusted while the others remain unchanged. In general, DHNN evolves faster in the synchronous update mode, but it is easy to fall into an infinite loop, and the opposite is true in asynchronous update mode. Example 1 shows the detailed steps of the above two update modes.

Example 1: Let us consider a DHNN with the same transition architecture illustrated in Fig. 2. The weight matrix and bias vector are $\boldsymbol{W}$, $\boldsymbol{b}$, respectively, where

$$\begin{aligned} \boldsymbol{W} &= \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & 0.2 \\ -0.5 & 0 & 0.6 \\ 0.2 & 0.6 & 0 \end{bmatrix} \\ \boldsymbol{b} &= [b_1, b_2, b_3] = [-0.1, 0, 0.1]. \end{aligned} \tag{5}$$

Let the initial state of DHNN be $\boldsymbol{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\} = \{1, -1, 1\}$. Then, for the two update methods, the next state of the network is as follows:

In the synchronous update mode, the next state is $\boldsymbol{x}^{(1)} = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\}$, where

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = f\left(\mathbf{W} \cdot \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}. \tag{6}$$

In the asynchronous update mode, we randomly select a neuron to update its state, assuming $x_1^{(0)}$ is selected, then the next state $\boldsymbol{x}^{(1)} = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\}$, where

$$\begin{aligned} x_1^{(1)} &= f\Big(w_{11} \cdot x_1^{(0)} + w_{21} \cdot x_1^{(0)} + w_{31} \cdot x_1^{(0)} - b_1\Big) = -1 \\ x_2^{(1)} &= x_2^{(0)}, \; x_3^{(1)} = x_3^{(0)}. \end{aligned} \tag{7}$$

In [37], the concept of the energy function is introduced into the Hopfield neural network, which provides a reliable basis for judging the stability of network operation. The running process of DHNN is the evolution of states by dynamics. For an arbitrary initial state $\boldsymbol{x}^{(0)}$, it evolves in the way of energy reduction, and finally reaches a stable state $\boldsymbol{x}$. The stable state $\boldsymbol{x}$ is called the Hopfield network attractor which satisfies $\boldsymbol{x} = f(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{b})$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ are weight matrix and bias vector, respectively. The key of Hopfield as a decoder is how to effectively decode the continuous data $\boldsymbol{y}$ in *Definition 1* into discrete sequence $\boldsymbol{x}_i$. In this work, the sequence $\boldsymbol{x}_i$ is used as the attractor of the Hopfield network to solve this problem. Furthermore, we propose *Theorem 1* below to ensure the feasibility of Hopfield network decoding.

Theorem 1: Let $\boldsymbol{X}_P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_P\}$ be a sequence set, which consists of P binary sequences with same length N. Then in the synchronous update mode, the sequences $\boldsymbol{x}_p$ or $-\boldsymbol{x}_p$, $p = 1, 2, \ldots, P$, are the attractors of the DHNN with the weight matrix $\boldsymbol{W} = \sum_{p=1}^{P} [\boldsymbol{x}_p(\boldsymbol{x}_p)^T - \mathbf{I}]$ and no bias terms, that is

$$f\big(\boldsymbol{W}\boldsymbol{x}_p\big) = \begin{cases} \boldsymbol{x}_p, & (N - P) - (m_1 + m_2 + \cdots + m_P) \geq 0 \\ -\boldsymbol{x}_p, & (N - P) - (m_1 + m_2 + \cdots + m_P) < 0 \end{cases} \tag{8}$$

where $m_k = (x_k)^T x_p, \; k = 1, \ldots, p - 1, p + 1, \ldots, P$ and $f(\cdot)$ be the symbolic function.

Proof: Since each symbol of the sequence set $\mathbf{X}$ is 1 or -1, we have

$$(\boldsymbol{x}_k)^T\boldsymbol{x}_p = \begin{cases} m_k, & p \neq k \\ N, & p = k, \end{cases} \quad k = 1, 2, \ldots, N \tag{9}$$

where $-N \leq m_p \leq N$. Thus, we have

$$\begin{aligned} &\boldsymbol{W}\boldsymbol{x}_p \\ &= \sum_{k=1}^{P}\left[\boldsymbol{x}_k(\boldsymbol{x}_k)^T - I\right]\boldsymbol{x}_p = \sum_{k=1}^{P}\left[\boldsymbol{x}_k(\boldsymbol{x}_k)^T\boldsymbol{x}_p - \boldsymbol{x}_p\right] \\ &= \boldsymbol{x}_1(\boldsymbol{x}_1)^T\boldsymbol{x}_p + \cdots + \boldsymbol{x}_p\left(\boldsymbol{x}_p\right)^T\boldsymbol{x}_p + \cdots + \boldsymbol{x}_P(\boldsymbol{x}_P)^T\boldsymbol{x}_p \\ &\quad - P\boldsymbol{x}_p \\ &= (m_1\boldsymbol{x}_1 + m_2\boldsymbol{x}_2 + \cdots + m_P\boldsymbol{x}_P) + (N-P)\boldsymbol{x}_p. \end{aligned} \tag{10}$$

Since each symbol of sequences is 1 or -1, (8) can be rewritten as

$$\begin{aligned} &f\left(\boldsymbol{W}\boldsymbol{x}_p\right) \\ &= f\left[(m_1\boldsymbol{x}_1 + m_2\boldsymbol{x}_2 + \cdots + m_P\boldsymbol{x}_P) + (N-P)\boldsymbol{x}_p\right] \\ &= \operatorname{sgn}\left[(m_1\boldsymbol{x}_1 + m_2\boldsymbol{x}_2 + \cdots + m_P\boldsymbol{x}_P) + (N-P)\boldsymbol{x}_p\right] \\ &= \begin{cases} \boldsymbol{x}_p, & (N-P) - (m_1 + m_2 + \cdots + m_P) \geq 0 \\ -\boldsymbol{x}_p, & (N-P) - (m_1 + m_2 + \cdots + m_P) < 0. \end{cases} \end{aligned} \tag{11}$$

This completes the proof. ■

Remark 1: In the asynchronous update mode, the equation in Theorem 1 is written as

$$\begin{aligned} &f\left(\boldsymbol{W}\boldsymbol{x}_p\right) \\ &= \begin{cases} \boldsymbol{x}_p, & (N-P) - (m_1 + m_2 + \cdots + m_P) \geq 0 \\ \hat{\boldsymbol{x}}, & (N-P) - (m_1 + m_2 + \cdots + m_P) < 0 \end{cases} \end{aligned} \tag{12}$$

where $\hat{\boldsymbol{x}}$ does not belong to the sequence set $\boldsymbol{X}_P$, meaning that DHNN converges to attractors other than $\boldsymbol{X}_P$. It is easy to see that the smaller of P, the greater the probability that x_p is the attractor of DHNN.

If DHNN evolves in an asynchronous update mode, when weight matrix $\boldsymbol{W}$ is a symmetric matrix, the network will converge to an attractor. However, when the network evolves in a synchronous update mode, $\boldsymbol{W}$ must be a nonnegative definite symmetric matrix, otherwise the network may oscillate [39]. Therefore, in this work, we selected a suitable P through experiments and adopted a generalized asynchronous update mode. Specifically, DHNN randomly updates the state of k neurons every time it evolves to avoid the drawbacks of synchronous and asynchronous update modes.

In HpGAN, DHNN is designed through the weight $\boldsymbol{W}$, which is the continuous matrix that generated by the decoder with P suitable binary sequences. It can be guaranteed that the decoder can successfully map to the original binary sequences when DHNN converges to a stable state by *Theorem 1*. Example 2 visually shows a basic process of HpGAN.

Example 2: Consider a binary sequence set $\mathcal{C} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ which consists of M different sequences of the same length $N = 3$. Randomly select P sequences from $\mathcal{C}$ and encode them as continuous data $\boldsymbol{W}$, where we assume $P = 2, \boldsymbol{x}_1 = [1, 1, -1], \boldsymbol{x}_2 = [1, -1, 1]$. By *Definition 1* (no bias terms), we have

$$\boldsymbol{W} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{3} \\ 0 & -\frac{2}{3} & 0 \end{bmatrix} \tag{13}$$

then feed continuous data $\boldsymbol{W}$ as training data into GAN. When GAN reaches equilibrium, GAN generate a new continuous data $\boldsymbol{W}'$ which is similar to $\boldsymbol{W}$. We assume

$$\boldsymbol{W}' = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0 & -0.65 \\ 0 & -0.67 & 0 \end{bmatrix} \tag{14}$$

and exploit it as the weight matrix of DHNN with three neurons. Through the dynamic evolution of DHNN, we can know that for any initial state, when DHNN is stable, DHNN will always converge to $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$.

Remark 2: It is also feasible to design encoders and decoders with other neural networks, such as autoencoders. Two specific reasons for choosing Hopfield networks over other encoders/decoder for discretization are:

1) DHNN is easy to build and very convenient to update the training set. Specifically, the weights and biases of the DHNN can be designed with simple multiplication calculations, instead of long training.
2) DHNN has good interpretability with a guaranteed decoding rate.

Overall, in one iteration, there are three main steps.

1) All 2-D continuous data generated by the encoder are used as training data of GAN.
2) GAN model is trained until it reaches Nash equilibrium.
3) The generator generates continuous 2-D data and converts it into binary sequences by the decoder.

Since GAN can generate data with similar characteristics to training data, i.e., $|\mathcal{M}(\mathcal{C}_{\text{Train}}) - \mathcal{M}(\mathcal{C}_{\text{GAN}})| < \varepsilon$, where ε is a constant, $\mathcal{M}(\mathcal{C}_{\text{Train}})$ and $\mathcal{M}(\mathcal{C}_{\text{GAN}})$ are the metric of training data and generate data, respectively. In the next iteration, we update the initial binary sequences with the generated better ones. For example, to minimize $\mathcal{M}(\mathcal{C})$, we replace C_{T1} in the training set with C_{G1}, when $\mathcal{M}(\mathcal{C}_{\text{G1}}) < \mathcal{M}(\mathcal{C}_{\text{T1}})$, where $C_{\text{T1}} \in \mathcal{M}(\mathcal{C}_{\text{Train}})$ and $C_{\text{G1}} \in \mathcal{M}(\mathcal{C}_{\text{GAN}})$. In this way, the sequences in the training set get improved after every iteration, and intuitively the average metric $\bar{\mathcal{M}}(\mathcal{C})$ of the training set gradually decrease. Therefore, the sequences generated by GAN also gets improved after every iteration. The pseudocode for HpGAN is given in Algorithm 1.

In Sections III and IV, we demonstrate the searching capabilities of HpGAN in two applications: in Section III, we use HpGAN to search some complementary and near-complementary sequences which are not from the training set; in Section IV, we use HpGAN to search new phase-coded sequences for pulse compression radar systems. In the two applications, in order to highlight the advantages of discrete sequences generated by HpGAN, we compare the binary sequences generated by GAN and HpGAN, respectively.

```
Algorithm 1 HpGAN
Initialization:
    Generate a binary sequence set $\mathcal{S}$ as training dataset
    through existing algorithms or constructions.
    Initialize the parameters $\mathbf{W}$ and $\mathbf{b}$ of GAN.
while 1 do:
1: Generate the continuous training dataset $\mathcal{S}_{\text{con}}$ by the
    encoder.
2: Train GAN with the continuous dataset $\mathcal{S}_{\text{con}}$.
3: Generate continuous dataset $\mathcal{S}_{\text{GAN}}$ by the generator,
    until GAN reaches Nash equilibrium.
4: Transform $\mathcal{S}_{\text{GAN}}$ into binary sequences $\mathcal{S}_{\text{bin}}$ by the
    decoder.
5: Compute metric $\mathcal{M}(\mathcal{C}_{\text{GAN}})$, and select $M$ sequences with
    good $\mathcal{M}(\mathcal{C})$ to update a new dataset $\mathcal{S}_{\text{new}}$, where $\mathcal{C}_{\text{GAN}} \in \mathcal{S}_{\text{bin}}$.
6: Update dataset, i.e., $\mathcal{S} \leftarrow \mathcal{S}_{\text{new}}$.
end
```

III. HpGAN for Complementary and Near-Complementary Sequences

Golay complementary pairs (GCPs) were proposed by Golay in his work on infrared spectrometry in 1949 [40], where their mathematical properties were studied in 1961 [41]. The existing known binary GCPs only have even-lengths in the form of $2^{\alpha} \cdot 10^{\beta} \cdot 26^{\gamma}$ where α, β, γ are nonnegative integers satisfying $\alpha + \beta + \gamma \geq 1$ [42], [43]. Motivated by the limited admissible lengths of binary GCPs, Fan *et al.* [44] proposed "Z-complementary pair (ZCP)" which features zero aperiodic auto-correlation sums for certain out-of-phase time-shifts around the in-phase position. Such a region is called a zero-correlation zone (ZCZ). Up to date, GCPs and odd-length binary ZCPs (OB-ZCPs) have found numerous applications in wireless engineering such as radar sensing [45], channel estimation [46], synchronization in 3G standard [47], and PMEPR reduction [3], [48]. The concept of complementary pairs was extended later to mutually orthogonal complementary sequence set (MOCSS) [49], which are widely used in MC-CDMA systems to eliminate multipath interference and multiuser interference [50], [51].

Let us consider a MOCSS $\mathcal{C} = \{\boldsymbol{c}_j^m(n) : j = 0, 1, \ldots, J-1; m = 0, 1, \ldots, M-1; n = 0, 1, \ldots, N-1\}$ which have zero auto- and cross correlation sum properties for all nontrivial nonzero time shifts. Specifically.

1) *Ideal Aperiodic Auto-Correlation Function (AAF)*: For the M sequences assigned to a user j, i.e., $\{c_j^m : m = 0, 1, \ldots, M-1\}$, the sum of the AAF of these sequences is zero for any nonzero shift
$$\text{AAF}_j(\tau) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1-\tau} c_j^m(n) c_j^m(n+\tau) = 0 \quad (15)$$
where delay $\tau = -N+1, \ldots, N-1, \tau \neq 0$. Any sequence in this set is called a complementary set sequence (CSS). In particular, when $M = 2$, the set is called a GCP, and any constituent sequence in this pair is called a Golay sequence (GS) [49].

2) Ideal Aperiodic Cross Correlation Function (ACF): For two flocks of complementary codes assigned to users j_1 and j_2, i.e., $\{\boldsymbol{c}_{j_1}^m, \boldsymbol{c}_{j_2}^m : m = 0, 1, \ldots, M-1\}$, the sum of their ACFs is always zero irrespective of the relative shift
$$\text{ACF}_{j_1,j_2}(\tau) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1-\tau} c_{j_1}^m(n) c_{j_2}^m(n+\tau) = 0 \quad (16)$$
where $\tau = -N+1, \ldots, N-1$ and $j_1 \neq j_2$.

Some known constructions of MOCSS are available in [52]. In Section III-A, we make use of HpGAN to search MOCSS. Our goal is to investigate and evaluate the capability of HpGAN, i.e., whether it can search for some MOCSSs which are not in the training dataset.

A. HpGAN for MOCSSs

In this section, we use HpGAN to search MOCSSs. As mentioned above, a MOCSS should satisfy (15) and (16) at the same time. Hence, we consider the following metric function $\mathcal{M}(\mathcal{C})$ which is the sum of all the nontrivial aperiodic auto- and cross correlation squares of a sequence set C:

1) Metric Function: For a binary sequence set $\mathcal{C} = \{\boldsymbol{x}_j^m(n) : j = 0, 1, \ldots, J-1; m = 1, 2, \ldots, M-1; n = 0, 1, \ldots, N-1; x_j^m(n) \in \{-1, 1\}\}$ consisting of MJ sequences of same length N, the metric function $\mathcal{M}(\mathcal{C})$ is defined below
$$\mathcal{M}(\mathcal{C}) = \sum_{j=0}^{J-1} \sum_{\tau=-N+1}^{N-1} |\text{AAF}_j(\tau)| + \sum_{j_1=0}^{J-1} \sum_{j_2=j_1+1}^{J-1} \sum_{\tau=-N+1}^{N-1} |\text{ACF}_{j_1,j_2}(\tau)|. \quad (17)$$

For MOCSSs, it is desired to have $\mathcal{M}^* = \inf \mathcal{M}(\mathcal{C}) = 0$. Our goal is to demonstrate the effectiveness of HpGAN by generating new MOCSSs.

2) Training Dataset: Generating sufficient training data is a prerequisite of using HpGAN, and the quality of the training sequence sets determines the performance of HpGAN. In this article, we exploit some known constructions in [12] to generate a training datasets $\mathcal{S}_{\text{train}}$, which contains 200 different MOCSSs, where $\mathcal{M}(\mathcal{C}) = 0$, $\mathcal{C} \in \mathcal{S}_{\text{train}}$ and $J = 2$, $M = 2, N = 8$ for each $\mathcal{C}$.

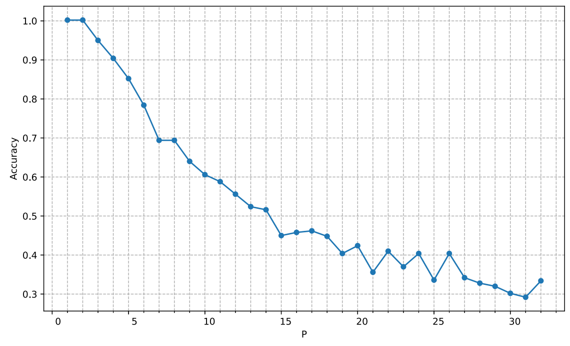
3) Encoder: We regard each sequence set $\mathcal{C}$ as a sequence of length 32. Based on *Definition 1*, we design an encoder to convert discrete data into continuous data. Specifically, we generated 500 continuous data as the training dataset of GAN by (1) with random P and $\mathbf{b}$. In general, the larger of P and $\mathbf{b}$, the more samples can be generated, but the lower the decoding accuracy. From Fig. 3, we can see that there is a decoding accuracy when both P and $\mathbf{b}$ take relatively small numbers. Therefore, considering the diversity of generated data and the decoding accuracy, we set $P \leq 4$ and $\mathbf{b} \in [0, 0.4]$ in HpGAN.

4) GAN: Both the generative model G and the discriminant model D in HpGAN are implemented by multilayer perceptrons. Each perceptron contains three layers of neurons: input layer, hidden layer, and output layer. The input of the generator

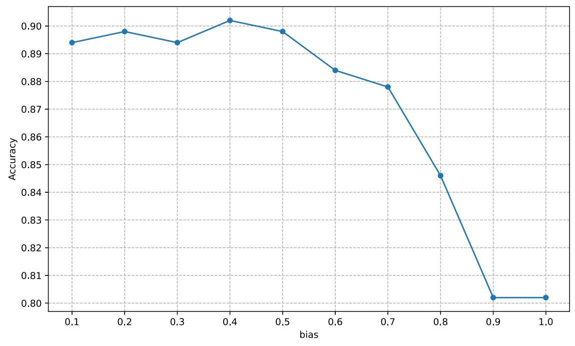
TABLE I

PARAMETERS OF HPGAN

Items	Parameters	Definition
Generative Model	$I_G = 100$	Number of neurons in the input layer
	$H_G = 1024$	Number of neurons in the hidden layer
	$O_G = 1024$	Number of neurons in the output layer
Discriminant Model	$I_D = 1024$	Number of neurons in the input layer
	$H_D = 1024$	Number of neurons in the hidden layer
	$O_D = 1$	Number of neurons in the output layer
Generative Model/Discriminant Model	$\alpha = 0.0001$	Learning rate of the generative/discriminant model
	batch $= 100$	Mini-batch size



(a)



(b)

Fig. 3. Illustration of the decoding accuracy with the two parameters P and bias. (a) Decoding accuracy with the change of parameter P. (b) Decoding accuracy with the change of parameter $\mathbf{b}$.

is a random noise vector $\mathbf{z}$, where $\mathbf{z}$ obeys uniform distribution, i.e., $\mathbf{z} \sim U(-1, 1)$. In the process of training GAN, the minibatch size was set to 100, and we randomly sampled five minibatches without replacement from the 500 continuous data to train the GAN. The parameter settings of HpGAN are listed in Table I. O_G and I_D were set to 1024. Since the perceptron model can only process 1-D data, we converted all training data into 1-D data with a length of 32 × 32 and fed them into HpGAN. The discriminator is equivalent to a two-class model, thus O_D was set to 1. Other parameters may be modified according to the effectiveness of the model.

5) Decoder: The decoder is a DHNN with 32 neurons which uses the generated data as its weight matrix and then decodes the generated data through the dynamic evolution of DHNN. We employ asynchronous updates to evolve DHNN. Specifically, during each update of DHNN, only five neurons are randomly selected to update. For example, when only one neuron i is updated, the evolution formula of DHNN is as follows:

$$x_j(t+1) = \begin{cases} \operatorname{sgn}\big(\text{net}_j(t)\big), & j = i \\ x_j(t), & j \neq i, \end{cases} \quad j = 1, 2, \ldots, n. \tag{18}$$

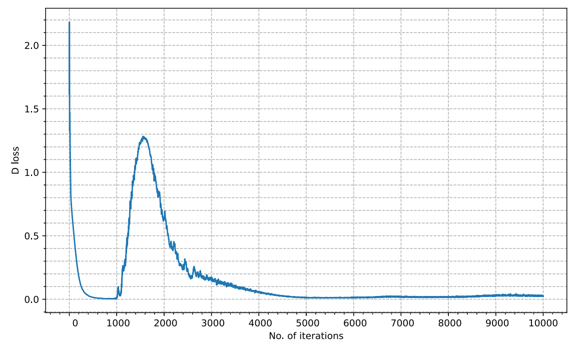
By this method, the search space can be increased, making it easier to get better results.

B. Performance Evaluation

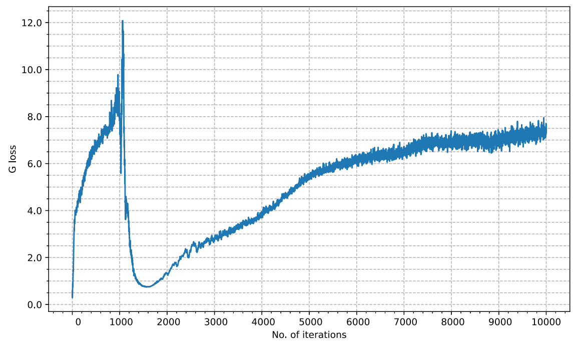
We evaluated the search performances of HpGAN on a PC with Intel Xeon W-2125 CPU at 4 GHz and 16 GB RAM. In this experiment, the number of iterations of the network was set to 10^4. In each iteration, we calculated the loss values of the generative model and the discriminant model to observe the convergence of HpGAN. To monitor the evolution of HpGAN, we evaluated the search ability of HpGAN every 100 iterations by calculating the metric $\mathcal{M}(\mathcal{C})$ of 100 sequence sets generated by the current generator. We exploited the minimum metric min$[\mathcal{M}]$ and the mean metric $E[\mathcal{M}]$ in the generated sequence sets to reflect the search effect and search trend of HpGAN.

We have compared the performances of HpGAN and GAN in sequence generation tasks to demonstrate the necessity of encoding in HpGAN. We have used the MOCSSs as training data and fed it to GAN without encoding, where GAN and HpGAN have the same structure. The loss values of the generator and discriminator of HpGAN and GAN are shown in Figs. 4 and 5, respectively. It can be seen from these two figures that after 2500 iterations, the loss values of the HpGAN discriminator gradually decrease and converge, and the loss values of the generator gradually become flat. However, the loss values of GAN do not converge, but exhibit an exponential growth trend.

Next, we compare the sequence set generation effects of HpGAN, SeqGAN, and GAN, as shown in Fig. 6, where triangles and circles, respectively, represent the min$[\mathcal{M}]$ and $E[\mathcal{M}]$ in the generated 100 sequence sets; the blue, green, and red lines represent the sequence set generation process of GAN, SeqGAN, and HpGAN, respectively. As shown in Fig. 6, after 2500 iterations, the min$[\mathcal{M}]$ of HpGAN reaches the optimal value, and its $E[\mathcal{M}]$ shows a decreasing trend with network iterations. However, the min$[\mathcal{M}]$ and $E[\mathcal{M}]$ of the sequence sets generated by GAN are not significantly improved. This shows that it is difficult for GAN to learn the effective features of the training sequences, when GAN is used to directly train binary sequences without encoding. Similar to GAN, with the continuous training of SeqGAN, the min$[\mathcal{M}]$

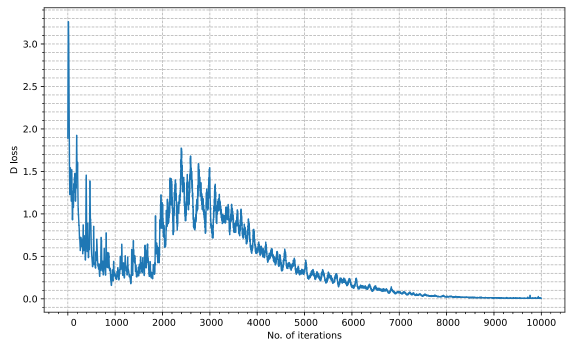


(a)

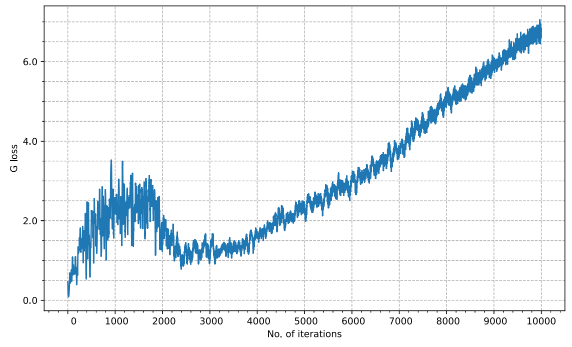


(b)

Fig. 4. Loss values of the generator and the discriminator of HpGAN with the change of the number of iterations. After 2500 iterations, the values of the generative model and the discriminant model tend to be flat. (a) Loss values of the discriminator with the change of the number of iterations. (b) Loss values of the generator with the change of the number of iterations.



(a)



(b)

Fig. 5. Loss values of the generator and discriminator of GAN. During the training process, the loss values of the discriminator gradually approaches zero, while the loss values of the generator show a rapid increase. (a) Loss values of the discriminator with the change of the number of iterations. (b) Loss values of the generator with the change of the number of iterations.

and $E[\mathcal{M}]$ of the generated sequence set fluctuate within a certain range. Although the effect of SeqGAN in generating sequence sets is better than that of GAN, it can also be seen that it has not learned the effective features of the sequence at present. Over the course of training, HpGAN generated 10 000 sequence sets, from which we found 119 different MOCSSs with $\mathcal{M}(\mathcal{C}) = 0$, and none of these sequence sets belong to the training sequence sets.

It is worth noting that the purpose of HpGAN is to train the generator to learn the probability distribution function $P_{\mathcal{C}}$ of the training data $\mathcal{C}$ so that it can generate data with similar characteristics to the training data. Once the training is completed, we can directly use the generator to quickly generate a large amount of new data without retraining. To prevent the problem of mode collapse in HpGAN, during the training of HpGAN, we denote the number of different sequence sets in the 100 sequence sets generated each time as d. From Fig. 7, the diversity of the generated sequence will decrease as the network continues to be trained, but after the network is trained 10 000 times, the difference rate of the HpGAN generated sequence set is still above 50%.

C. HpGAN for Optimal and Suboptimal OB-ZCPs

In this section, to further demonstrate that the sequences generated by HpGAN may not be produced by systematic construction, we search optimal and suboptimal OB-ZCPs by HpGAN. In this experiment, we constructed 128 of OB-ZCPs

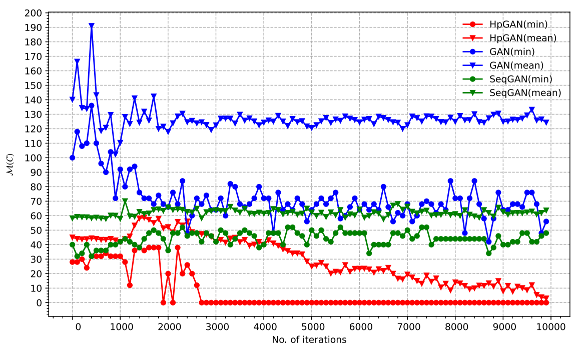


Fig. 6. Comparison among the min$[\mathcal{M}]$ and $E[\mathcal{M}]$ of the sequence set obtained through GAN, SeqGAN, and HpGAN.

of length 15 by the approach in [53] as the training data. This was done by deleting the first or last element of the binary GCPs of length 16. In the training data, the maximum ZCZ length of all OB-ZCPs is 4. In addition, the minimum PMEPR (interested readers may refer to [3], [48], [53] and references therein for more details) of each sequence in the training data is 1.7272. We designed HpGAN whose network structure and parameters are the same as those of HpGAN when searching for MOCSSs in Section III-A.

After the experiment, we found 140 OB-ZCPs, where none of them belong to the training data. In particular, an optimal Type-I OB-ZCP $s_{\text{Type-I}}$, Type-II OB-ZCP $s_{\text{Type-II}}$ and

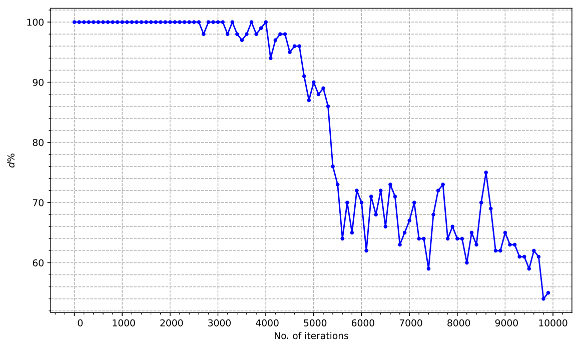


Fig. 7. In the training of HpGAN, the number of sequence sets that are different from each other in 100 sequence sets generated by HpGAN each 100 iterations.

suboptimal OB-ZCP $s_{\text{suboptimal}}$ found by HbGAN as shown at the bottom of the page.

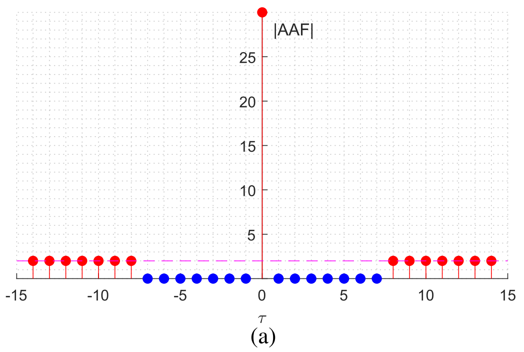
A plot of their individual aperiodic auto-correlation sum magnitudes is shown in Fig. 8. One can see that the $s_{\text{Type-I}}$ and $s_{\text{Type-II}}$ are optimal. A suboptimal OB-ZCP $s_{\text{suboptimal}}$ found by HpGAN is shown in Fig. 9. HpGAN also found the sequence with PMEPR metric PMEPR$(s_{\text{PMEPR}}) \approx 1.6667$, as shown at the bottom of the next page, which is better than any sequence in the training data.

IV. HpGAN for Pulse Compression Radar

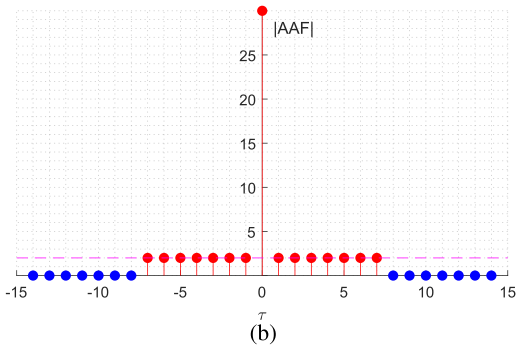
In modern complex applications, the radar is required to have a large detection range and high resolution. However, for a pulsed radar system that transmits a fixed carrier frequency, its resolution is inversely proportional to the transmitted pulsewidth. Thus, there is a tradeoff between distance and resolution. Pulse compression radar can take into account the detection range and resolution at the same time by modulated pulses [30], [54], [55]. The key is to use modulated pulses (e.g., phase-coded pulse) rather than conventional nonmodulated pulses.

Let $\boldsymbol{s}$ be a binary probing sequence of length N, and $\boldsymbol{y}$ denotes the received sequence of length N. Let $\{h_n\}_{n=-N+1,n\neq 0}^{N-1}$ denote the corresponding amplitude coefficients for the adjacent range bins or clutter patches. $\mathbf{J}_n$ denote a shift matrix that takes into account the fact which the clutter returns from adjacent range bins need different propagation times to reach the radar receiver [55]

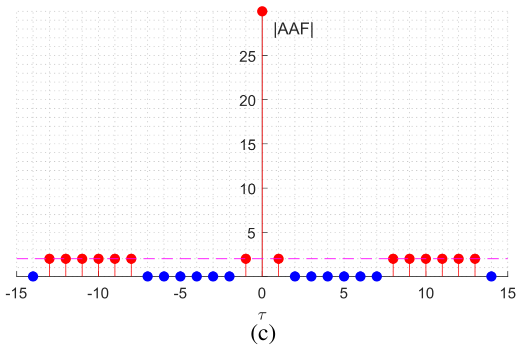
$$\mathbf{J}_n = \begin{bmatrix} \overbrace{0 \quad \cdots \quad 0}^{n} & 1 & & \mathbf{0} \\ & & \ddots & \\ & & & 1 \\ \mathbf{0} & & & \end{bmatrix}_{N\times N} = \mathbf{J}_{-n}^T, \quad n = 1, 2, \ldots, N-1. \tag{19}$$



(a)



(b)



(c)

Fig. 8. AAF sum magnitudes of OB-ZCP $s_{\text{Type-I}}$, OB-ZCP $s_{\text{Type-II}}$ and suboptimal OB-ZCP $s_{\text{suboptimal}}$, respectively. (a) For Type-I OB-ZCP $s_{\text{Type-I}}$. (b) For Type-II OB-ZCP $s_{\text{Type-II}}$. (c) For suboptimal OB-ZCP $s_{\text{suboptimal}}$.

Following the definition in [55] and [56], after subpulse-matched filtering (MF) and analog-to-digital conversion, we can write the received sequence $\boldsymbol{y}$ as:

$$\boldsymbol{y} = h_0\boldsymbol{s} + \sum_{n=1-N,n\neq 0}^{N-1} h_n \mathbf{J}_n \boldsymbol{s} + \boldsymbol{w} \tag{20}$$

where $\boldsymbol{w}$ denotes the additive white Gaussian noise (AWGN).

Given the received sequence $\boldsymbol{y}$, the radar's objective is to estimate h_0, where h_0 corresponds to the range bin of interest. To this end, the operation of a conventional receiver

$$s_{\text{Type-I}} = \begin{bmatrix} -1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 \\ -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 & +1 & +1 \end{bmatrix}$$

$$s_{\text{Type-II}} = \begin{bmatrix} -1 & +1 & -1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 & +1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 \end{bmatrix}$$

$$s_{\text{suboptimal}} = \begin{bmatrix} -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & +1 & -1 \\ -1 & -1 & +1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 \end{bmatrix}.$$

is described by the following equation [38], [55]–[59]:

$$\hat{h}_0 = \frac{\boldsymbol{s}^T \boldsymbol{y}}{\boldsymbol{s}^T \boldsymbol{s}} = h_0 + \sum_{n=1-N, n\neq 0}^{N-1} h_n \frac{\boldsymbol{s}^T \mathbf{J}_n \boldsymbol{s}}{\boldsymbol{s}^T \boldsymbol{s}} \tag{21}$$

where the AWGN is ignored since the received signal is interference-limited. Because h_0 and $\{h_n\}_{n=-NN+1,n\neq 0}^{N-1}$ are unknown, it is reasonable to define the signal-to-interference ratio (SIR) Υ_{MF} at the output of the above receiver is as follows:

$$\Upsilon_{\mathrm{MF}} = \frac{\left(\boldsymbol{s}^T \boldsymbol{s}\right)^2}{\sum_{n=1-N,n\neq 0}^{N-1}\left(\boldsymbol{s}^T \mathbf{J}_n \boldsymbol{s}\right)^2}. \tag{22}$$

There is a rich body of literature on maximizing Υ_{MF} over the set of binary sequences, which is referred to as the well-known "merit factor problem" [57]–[61]. For such sequences, the best-known merit factor of 14.08 is achieved by the Barker sequence of length 13 [62].

In the presence of Gaussian white noise, the above MF estimator can provide the largest signal-to-noise ratio (SNR). However, some clutters may cause interference to the received information, especially in the scene of weak target detection. Hence, interference suppression is important. This motivates the design of mismatched filter (MMF) estimators to suppress the interference of clutter [55], [56], [63].

The MMF estimator uses a general real-valued sequence $\boldsymbol{x}$ instead of the phase sequence $\boldsymbol{s}$, and correlates the received sequence, giving

$$\hat{h}_0 = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\boldsymbol{x}^T \boldsymbol{s}} = h_0 + \sum_{n=1-N, n\neq 0}^{N-1} h_n \frac{\boldsymbol{x}^T \mathbf{J}_n \boldsymbol{s}}{\boldsymbol{x}^T \boldsymbol{s}}. \tag{23}$$

The receiver optimizes $\boldsymbol{x}$ by maximizing the following formula:

$$\Upsilon_{\mathrm{MMF}} = \frac{\left(\boldsymbol{x}^T \boldsymbol{s}\right)^2}{\sum_{n=1-N,n\neq 0}^{N-1}\left(\boldsymbol{x}^T \mathbf{J}_n \boldsymbol{s}\right)^2}. \tag{24}$$

It has been shown in [56] that, given a phase code $\boldsymbol{s}$, the optimal sequence $\boldsymbol{x}$ that maximizes Υ_{MMF} is $\boldsymbol{x}^* = \mathbf{R}^{-1}\boldsymbol{s}$, where matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \sum_{n=1-N,n\neq 0}^{N-1} \mathbf{J}_n \boldsymbol{s}\boldsymbol{s}^T \mathbf{J}_n^T. \tag{25}$$

Substituting $\boldsymbol{x}^* = \mathbf{R}^{-1}\boldsymbol{s}$ into (24), we have

$$\Upsilon_{\mathrm{MMF}} = \boldsymbol{s}^T \mathbf{R}^{-1} \boldsymbol{s}. \tag{26}$$

Note that Υ_{MMF} only depends on the phase code $\boldsymbol{s}$. Hence, the objective for the design of the MMF estimator is then to discover a phase-code s that can maximize Υ_{MMF} in (26).

A. HpGAN for Pulse Compression Radar

In this section, we use HpGAN to search the phase sequences for pulse compression radar which maximize the metric $\mathcal{M}(\boldsymbol{s})$. The metric $\mathcal{M}(\boldsymbol{s})$ is defined as

$$\mathcal{M}(\boldsymbol{s}) = \boldsymbol{s}^T \mathbf{R}^{-1} \boldsymbol{s} \tag{27}$$

where matrix $\mathbf{R}$ is given by (25).

Massive training data is a necessary condition for HpGAN. Unfortunately, there are no suitable mathematical constructions that can generate a large number of sequences with good metric $\mathcal{M}(\boldsymbol{s})$ with an identical length. For this, we employed a genetic algorithm (GA) to generate 250 different sequences as the training sequences of HpGAN. In the GA, we used random sequences as parents and then searched for sequences through crossover, mutation, and selection operations where the fitness function is the metric $\mathcal{M}(\boldsymbol{s})$. For the MMF estimator, these sequences which are generated by GA yielded SIR $\Upsilon_{\mathrm{MMF}} \in [10, 21]$.

For this application, the main architecture of HpGAN is similar to that of the network in Section III. Major differences in the specific design and parameter selection are summarized as follows.

1) In the encoder, we aim to increase the diversity of encoded sequences while ensuring high decoding accuracy. Therefore, the number of sample sequences in definition 1 is set to $P \leq 8$, because of the larger sequence length than that in Section III, whilst the bias vector is still set to $\mathbf{b} \in [0, 0.4]$.
2) Since we need to convert the encoded 2-D sequences into 1-D and then feed it into the GAN, the length of the training sequences is 59×59. A simple perceptron model is difficult to effectively extract the characteristics of the training data. Therefore, we use a multilayer perceptron as the generative model and discriminant model in HpGAN, in which each perceptron contains two hidden layers in this experiment. The parameters settings are listed in Table II.
3) In this experiment, our goal is not to generate sequences similar to the initial training sequences but to generate sequences that are better than the initial sequences. However, training the network with only initial sequences cannot achieve our goal, because the principle of GAN is to generate data with similar characteristics to the training data. In this regard, the sequences that we generate during network training are better than the initial sequences, as new training sequences update the initial sequences. Moreover, after every 2500 iterations (according to our experience, the model reaches equilibrium after 2500 iterations), we exploit the current model to generate a new sequence set $\mathcal{S}_{\mathrm{new}}$ to update the initial training sequence set $\mathcal{S}_{\mathrm{init}}$, i.e., $\mathcal{S}_{\mathrm{init}} \leftarrow \mathcal{S}_{\mathrm{new}}$.

$$s_{\mathrm{PMEPR}} = \left[-1\ -1\ -1\ +1\ +1\ +1\ -1\ +1\ -1\ +1\ -1\ -1\ +1\ -1\ -1\right]$$

TABLE II

PARAMETERS OF HPGAN

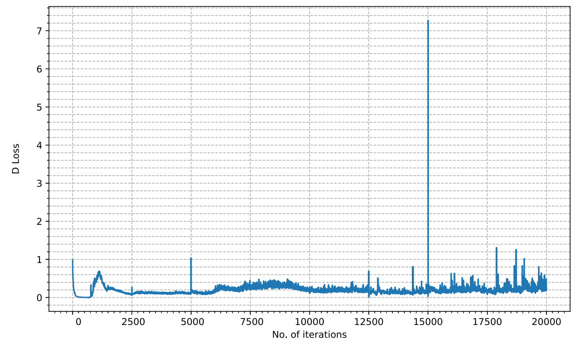
Items	Parameters	Definition
Generative Model	$I_G = 100$	Number of neurons in the input layer
	$H_G^1 = 1024$	Number of neurons in the first hidden layer
	$H_G^2 = 1024$	Number of neurons in the second hidden layer
	$O_G = 3481$	Number of neurons in the output layer
Discriminant Model	$I_D = 3481$	Number of neurons in the input layer
	$H_D^1 = 1024$	Number of neurons in the first hidden layer
	$H_D^2 = 1024$	Number of neurons in the second hidden layer
	$O_D = 1$	Number of neurons in the output layer
Generative Model/Discriminant Model	$\alpha = 0.0001$	Learning rate of the generative/discriminant model
	batch $= 100$	Mini-batch size

B. Performance Evaluation

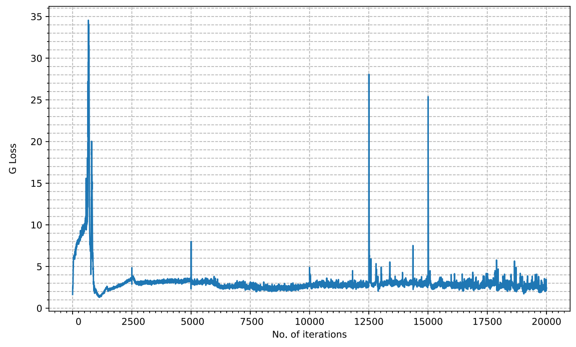
In the experiment, HpGAN was executed 2×10^4 iterations to train the model in order to generate better sequences. According to our testing experience, GAN reached equilibrium after about 2500 iterations. Furthermore, after 2500 iterations of GAN, GAN can generate sequences with similar characteristics to the training sequences. Therefore, we updated the training set with the sequences generated by GAN after every 2500 iterations and train the network again. When the training set is updated for the kth time, we exploit the current generative model to generate a large amount of data, and select 100 different sequences as the new training set $\mathcal{S}_{\text{new}}$. Since the mean metric of the sequences in the initial training set $\mathcal{S}_{\text{init}}$ is 13 and we also consider the learning ability of HpGAN, the metric of sequence in the new training is set to $\mathcal{M}(\boldsymbol{s}) \geq 13+3\times k$, $\boldsymbol{s} \in \mathcal{S}_{\text{new}}$. After each update of the training set, GAN learns the sequences in $\mathcal{S}_{\text{new}}$, and the average metric of the generated sequences is $\bar{\mathcal{M}}(\boldsymbol{s}') \approx \mathcal{M}(\boldsymbol{s})$, where $\boldsymbol{s}'$ is the sequence generated by the current GAN. As in Section III, we calculate the loss values of the generative model and the discriminant model to observe the convergence of the network. To monitor the evolution of HpGAN, every 100 iterations, we evaluated the searching capability of HpGAN and record their mean metric $E[\mathcal{M}]$ and maximum metric $\max[\mathcal{M}]$. We exploit the minimum metric $\max[\mathcal{M}]$ and the mean metric $E[\mathcal{M}]$ in the generated sequence sets to reflect the search effect and search trend of HpGAN.

From Fig. 9, we can see that after 2500 iterations, the loss functions of HpGAN gradually converge. Each time the training set is updated, the loss functions of the model fluctuate and then gradually stabilize showing how the model makes adjustment after the training set is updated.

The evolution curves of the mean metric $E(\mathcal{M})$ and maximum metric $\max(\mathcal{M})$ with respect to the number of iterations are shown in Fig. 10. We can see that the overall trends of $E(\mathcal{M})$ and $\max(\mathcal{M})$ is gradually increase with the number of iterations, especially after updating the training set, both $E(\mathcal{M})$ and $\max(\mathcal{M})$ get greatly improved. For the MMF estimator, the well-known Legendre sequence yields $\mathcal{M}(\boldsymbol{s}_L) \approx 10.98$, which is represented by the brown dashed line in the figure. The green and purple dashed lines represent the average metric $E(\mathcal{M})_{\text{init}} \approx 13.34$ and the maximum metric $\max(\mathcal{M})_{\text{init}} \approx 23.25$ of the initial training set, respectively. These sequences are all obtained by GA and are better than the Legendre sequence, among which the best sequence is denoted as $\boldsymbol{s}_{\text{GA}}$. To compare with the performance of AlphaSeq, we also marked the metric $\mathcal{M}(\boldsymbol{s}_{\text{Alpha}}) \approx 33.45$ of the best sequence $\boldsymbol{s}_{\text{Alpha}}$ obtained by AlphaSeq with a yellow dashed line.



(a)



(b)

Fig. 9. Loss values of the generator and discriminator of HpGAN. Each time the training set is updated, the loss functions of the model fluctuate and then gradually stabilize. (a) Loss values of the discriminator of HpGAN. (b) Loss values of the generator of HpGAN.

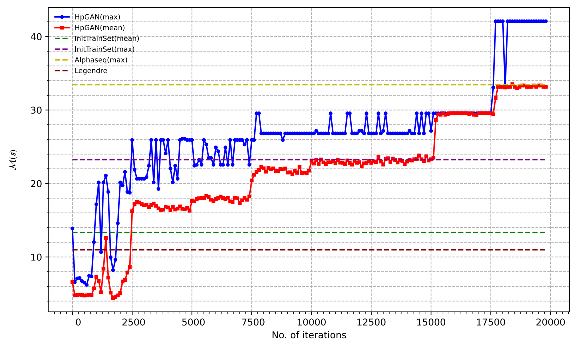


Fig. 10. Training process of HpGAN to search a phase-coded sequence for pulse compression radar with the evolution curves of the mean metric $E(\mathcal{M})$ and the maximum metric $\max(\mathcal{M})$.

It can be seen from Fig. 10 that after 2500 iterations of training the network with the initial training data, the network can generate some sequences which are better than that from the training set, but the mean value of the generated sequences is lower than $E(\mathcal{M})_{\text{init}}$. After the training set is updated for the first time, the sequences generated by HpGAN are better than the initial training set as a whole, and as the training set is updated again, the metric of the sequences generated by the network gets improved again. In particular, after 17 500 iterations, HpGAN has found sequences whose metrics are better than $\max(\mathcal{M})_{\text{Alpha}}$, and the mean metric of HpGAN generated sequences is close to $\max(\mathcal{M})_{\text{Alpha}}$. This figure also demonstrates that HpGAN is an effective search tool, which can continuously improve the search ability as the training set is updated.

When HpGAN training is completed, we can exploit the generator to quickly generate sequences with linear complexity. After the 20 000th iteration, we exploit the generator searches a sequence with metric $\mathcal{M}(\boldsymbol{s}_{\text{HpGAN}}) \approx 42.07$

$$\begin{aligned}&\boldsymbol{s}_{\text{HpGAN}}\\&= [1010101010101010110101101011001101001111\\&\quad 0000011111111111111].\end{aligned}$$

Compared to the well-known Legendre sequence, $\boldsymbol{s}_{\text{HpGAN}}$ increases the SNR of the MMF estimator in the pulse compression radar system by four times. Moreover, compared to $\boldsymbol{s}_{\text{Alpha}}$ and $\boldsymbol{s}_{\text{GA}}$, $\boldsymbol{s}_{\text{HpGAN}}$ improves the SIR by 8.62 and 18.82 at output of an MMF estimator, respectively.

V. Conclusion

In this article, we proposed a novel algorithm for searching sequences based on GAN, which is called HpGAN. In the HpGAN training process, since the network is updated inversely according to the loss function, the calculation of the metric function of the sequence is avoided. As a result, the computational complexity is reduced, making HpGAN more applicable to many different application scenarios, especially for those with complex metric functions.

We demonstrated the search ability of HpGAN through two applications. In the first application, we successfully found MOCSSs and optimal/suboptimal OB-ZCPs, showing that HpGAN is capable of achieving global optimal solutions and can obtain more results based on existing tools. In the second application, based on the training set generated by the GA algorithm, HpGAN found a new sequence, which is far superior to the existing sequences for increasing the SNR of the MMF estimator in the pulse compression radar system. Compared with the sequence discovered by AlphaSeq, the sequence found by HpGAN improves the SIR by 8.62 at the output of an MMF estimator.

As future work, more efficient GAN architecture for sequences may be designed. For example, it is likely to use deep CNNs or residual neural networks to build generative models and discriminative models so as to better extract the features of sequences. Designing a more suitable encoder and decoder is an interesting research problem for searching longer sequences.

References

[1] S. Mertens, "Exhaustive search for low-autocorrelation binary sequences," *J. Phys. A, Math. Gen.*, vol. 29, no. 18, pp. 473–481, 1996.

[2] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, vol. 122. Reading, MA, USA: Addison-Wesley, 1995.

[3] J. A. Davis and J. Jedwab, "Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed–Muller codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2397–2417, Nov. 1999.

[4] D. Pham and D. Karaboga, *Intelligent Optimisation Techniques: Genetic Algorithms Tabu Search Simulated Annealing and Neural Networks*. Springer, 2012.

[5] P. Stoica, H. He, and J. Li, "New algorithms for designing unimodular sequences with good correlation properties," *IEEE Trans. Signal Process.*, vol. 57, no. 4, pp. 1415–1425, Apr. 2009.

[6] M. Soltanalian and P. Stoica, "Computational design of sequences with good correlation properties," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2180–2193, May 2012.

[7] J. Song, P. Babu, and D. P. Palomar, "Optimization methods for designing sequences with low autocorrelation sidelobes," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 3998–4009, Aug. 2015.

[8] M. A. Kerahroodi, A. Aubry, A. D. Maio, M. M. Naghsh, and M. Modarres-Hashemi, "A coordinate-descent framework to design low PSL/ISL sequences," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 5942–5956, Nov. 2017.

[9] H. Deng, "Synthesis of binary sequences with good autocorrelation and crosscorrelation properties by simulated annealing," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 32, no. 1, pp. 98–107, Jan. 1996.

[10] H. Deng, "Polyphase code design for orthogonal netted radar systems," *IEEE Trans. Signal Process.*, vol. 52, no. 11, pp. 3126–3135, Nov. 2004.

[11] X. Deng and P. Fan, "New binary sequences with good aperiodic autocorrelations obtained by evolutionary algorithm," *IEEE Commun. Lett.*, vol. 3, no. 10, pp. 288–290, Oct. 1999.

[12] W. H. Mow, K.-L. Du, and W. H. Wu, "New evolutionary search for long low autocorrelation binary sequences," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 1, pp. 290–303, Jan. 2015.

[13] F. Hu, P. Z. Fan, M. Darnell, and F. Jin, "Binary sequences with good aperiodic autocorrelation functions obtained by neural network search," *Electron. Lett.*, vol. 33, no. 8, pp. 688–689, Apr. 1997.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[16] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2016, pp. 223–228.

[17] M. Kim, W. Lee, and D.-H. Cho, "A novel PAPR reduction scheme for OFDM system based on deep learning," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 510–513, Mar. 2018.

[18] J. Kim, B. Lee, H. Lee, Y. Kim, and J. Lee, "Deep learning-assisted multi-dimensional modulation and resource mapping for advanced OFDM systems," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[19] A. Alhussain, H. Kurdi, and L. Altoaimy, "A neural network-based trust management system for edge devices in peer-to-peer networks," *Comput., Mater. Continua*, vol. 59, no. 3, pp. 805–816, 2019.

[20] M. Kim, N.-I. Kim, W. Lee, and D.-H. Cho, "Deep learning-aided SCMA," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 720–723, Apr. 2018.

[21] H. Kim, S. Oh, and P. Viswanath, "Physical layer communication via deep learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 5–18, May 2020.

[22] A. Maamar and K. Benahmed, "A hybrid model for anomalies detection in AMI system combining K-means clustering and deep neural network," *Comput., Mater. Continua*, vol. 60, no. 1, pp. 15–39, 2019.

[23] J. Wang, Y. Gao, W. Liu, W. Wu, and S.-J. Lim, "An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks," *Comput., Mater. Continua*, vol. 58, no. 3, pp. 711–725, 2019.

[24] H. Li, "Multiagent Q-learning for Aloha-like spectrum access in cognitive radio systems," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, no. 1, Dec. 2010, Art. no. 876216.

[25] L. R. Faganello, R. Kunst, C. B. Both, L. Z. Granville, and J. Rochol, "Improving reinforcement learning algorithms for dynamic spectrum allocation in cognitive sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 35–40.

[26] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, “Deep reinforcement learning for dynamic multichannel access in wireless networks,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[27] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, “Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, Apr. 2019.

[28] O. Naparstek and K. Cohen, “Deep multi-user reinforcement learning for distributed dynamic spectrum access,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[29] P. Liu, Y. Liu, T. Huang, Y. Lu, and X. Wang, “Decentralized automotive radar spectrum allocation to avoid mutual interference using reinforcement learning,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 1, pp. 190–205, Feb. 2021.

[30] Y. Shao, S. C. Liew, and T. Wang, “AlphaSeq: Sequence discovery with deep reinforcement learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3319–3333, Sep. 2020.

[31] J. Hu, Z. Wei, Y. Li, H. Li, and J. Wu, “Designing unimodular waveform(s) for MIMO radar by deep learning method,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 2, pp. 1184–1196, Apr. 2021.

[32] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Proc. NIPS*, 2014, pp. 2672–2680.

[33] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a Laplacian pyramid of adversarial networks,” in *Proc. NIPS*, Jan. 2015, pp. 1486–1494.

[34] F. Huszár, “How (not) to train your generative model: Scheduled sampling, likelihood, adversary?” 2015, *arXiv:1511.05101*.

[35] L. T. Yu, W. Zhang, J. Wang, and Y. Yu, “SeqGAN: Sequence generative adversarial nets with policy gradient,” in *Proc. AAAI*, Feb. 2017, pp. 2852–2858.

[36] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.

[37] J. J. Hopfield and D. W. Tank, “‘Neural’ computation of decision in optimisation problems,” *Biol. Cybern.*, vol. 52, pp. 141–152, Jul. 1985.

[38] M. Golay, “The merit factor of Legendre sequences (corresp.),” *IEEE Trans. Inf. Theory*, vol. IT-29, no. 6, pp. 934–936, Nov. 1983.

[39] L. Q. Han, *Artificial Neural Network Theory, Design and Application*. Beijing, China: Chemical Industry Press, 2007.

[40] M. J. E. Golay, “Multi-slit spectrometry,” *J. Opt. Soc. Amer.*, vol. 39, no. 6, pp. 437–444, 1949.

[41] M. Golay, “Complementary series,” *IRE Trans. Inf. Theory*, vol. 7, no. 2, pp. 82–87, Apr. 1961.

[42] P. Fan and M. Darnell, *Sequence Design for Communications Applications*. New York, NY, USA: Wiley, 1996.

[43] M. G. Parker, K. G. Paterson, and C. Tellambura, “Golay complementary sequences,” in *Wiley Encyclopedia of Telecommunications*, J. G. Proakis, Ed. New York, NY, USA: Wiley, 2002.

[44] P. Fan, W. Yuan, and Y. Tu, “Z-complementary binary sequences,” *IEEE Signal Process. Lett.*, vol. 14, no. 8, pp. 509–512, Aug. 2007.

[45] A. Pezeshki, A. R. Calderbank, W. Moran, and S. D. Howard, “Doppler resilient Golay complementary waveforms,” *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 4254–4266, Sep. 2008.

[46] H. M. Wang, X. Q. Gao, B. Jiang, X. H. You, and W. Hong, “Efficient MIMO channel estimation using complementary sequences,” *IET Commun.*, vol. 1, no. 5, pp. 962–969, Oct. 2007.

[47] B. M. Popović, “Method and apparatus for efficient synchronization in spread spectrum communications,” U.S. Patent 6 567 482 B1, May 20, 2003.

[48] Z. Liu and Y. L. Guan, “16-QAM almost-complementary sequences with low PMEPR,” *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 668–679, Feb. 2016.

[49] Z. Liu, Y. L. Guan, and W. H. Mow, “Asymptotically locally optimal weight vector design for a tighter correlation lower bound of quasi-complementary sequence sets,” *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3107–3119, Jun. 2017.

[50] Z. Liu, Y. L. Guan, and H. H. Chen, “Fractional-delay-resilient receiver design for interference-free MC-CDMA communications based on complete complementary codes,” *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1226–1236, Mar. 2015.

[51] S.-Y. Sun, H.-H. Chen, and W.-X. Meng, “A survey on complementary-coded MIMO CDMA wireless communications,” *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 52–69, 1st Quart., 2015.

[52] H. H. Chen, *The Next Generation CDMA Technologies*. Hoboken, NJ, USA: Wiley, 2007.

[53] Z. Liu, U. Parampalli, and Y. L. Guan, “Optimal odd-length binary Z-complementary pairs,” *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5768–5781, Sep. 2014.

[54] M. Shinriki and H. Takase, “Binary codes for multirange-resolution radar and pulse-compression properties,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 2, pp. 1549–1555, Apr. 2014.

[55] R. M. Davis, R. L. Facnte, and R. P. Perry, “Phase-coded waveforms for radar,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, pp. 401–408, Jan. 2007.

[56] P. Stoica, J. Li, and M. Xue, “On sequences with good correlation properties: A new perspective,” in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jul. 2007, pp. 1–5.

[57] M. Golay, “The merit factor of long low autocorrelation binary sequences,” *IEEE Trans. Inf. Theory*, vol. IT-28, no. 3, pp. 543–549, May 1982.

[58] T. Høholdt, “The merit factor problem for binary sequences,” in *Proc. Int. Symp. Appl. Algebra, Algebr. Algorithms, Error-Correcting Codes (AAECC)*. Berlin, Germany: Springer, 2006, pp. 51–59.

[59] J. Jedwab, “What can be used instead of a Barker sequence?” *Contemp. Math.*, vol. 461, pp. 153–178, Feb. 2008.

[60] J. Jedwab, “A survey of the merit factor problem for binary sequences,” in *Sequences and Their Applications—SETA* (Lecture Notes in Computer Science), vol. 3486, T. Helleseth, D. Sarwate, H. Y. Song, and K. Yang, Eds. Heidelberg, Germany: Springer, 2005, pp. 30–55.

[61] R. Ferguson and J. Knauer, “Optimization methods for binary sequences—The merit factor problem,” in *Proc. MITACS 6th Annu. Conf.* Calgary, AB, Canada: Univ. Calgary, May 2005.

[62] R. H. Barker, “Group synchronizing of binary digital systems,” in *Communication Theory*, W. Jackson, Ed. London, U.K.: Butterworths, 1953.

[63] M. J. Lindenfeld, “Mismatched filters for incoherent pulse compression in laser radar,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 2, pp. 1252–1260, Apr. 2021.