Received September 3, 2019, accepted September 15, 2019, date of publication September 24, 2019,
date of current version October 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943420

Real-Time Phase-Only Nulling Based on Deep Neural Network With Robustness

ZHONGHUI ZHAO, HUILING ZHAO, MINGXUAN ZHENG, AND JUNJIE TANG
School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China
Corresponding author: Zhonghui Zhao (zhaozhh@mail.nwpu.edu.cn)

ABSTRACT Phase-only nulling under sidelobe and mainlobe constraints is a problem of interest in array synthesis which is a nonlinear problem without analytical solution. To reduce the computational cost of phase-only array nulling on-line, this paper proposes a real-time phase-only array synthesis method based on the deep neural network. The on-line real-time prediction of element excitation phase is achieved by the trained neural network which can be done off-line. The performance of the trained neural network is related with the number of data. Firstly, in order to obtain a large enough database for the deep neural network efficiently, a multi-task phase-only array synthesis model with nulling operation and sidelobe control is relaxed to a convex problem and solved by direct iterative rank refinement. Then, the deep neural network is devised to emulate the phase array nulling behavior. This is carried out by the design of the structure of the network, the dataset structure and the loss function of the network. To validate the performance of the deep neural network, the phase-only nulling of 10-element and 16-element linear array based on the deep neural network is realized and tested. Experimental results demonstrate that the proposed real-time array synthesis method not only satisfies the desired array pattern property but also shows robustness to the array imperfections. Robustness is validated with Monte Carlo test.

INDEX TERMS Array pattern synthesis, deep neural network, interference nulling, robust array synthesis.

I. INTRODUCTION

Antenna array pattern synthesis is to generate the specific pattern by determining the amplitudes, phases or positions of the array elements, which is widely used in radar, sonar and communication systems [1]–[4]. In order to suppress the noises and interferences, the array pattern is designed with low sidelobe levels. Moreover, forming nulls at the interference directions is a more efficient anti-interference technique. Phase-only array synthesis has received much attention due to its economic and simplicity in the feeding network [5], [6]. Phase perturbation technique [5] enables the nulling equations to be linearized and then an analytical solution can be obtained. However, it is incapable of nulling symmetry jammers and will result in the increasing of sidelobe level. In order to achieve multiple goals in nulling, several evolution algorithms [7]–[9] have been used to solve the nonlinear phase-only problem, but it is quite time consuming since evolution algorithms converge slowly. To adapt to the changing environment, the array requires a rapid nulling response. Convex optimization has been applied in phase-only pattern synthesis problem after the relaxation operation [10], [11], and demonstrates that the method is more efficient than evolution algorithms. But the computation load of convex problem will increase with the number of the array elements and the number of the constraints. Furthermore, it is still far more beyond real-time response.

For the sake of response instantly to the changing environment with desired pattern, neural network based beamformer was applied to emulate the given array beamforming model [12], [13]. After learning the behavior from a certain number of the input-output pairs, the trained neural network with well generalization ability can predict the output under any inputs in the domain. The trained neural network can response in real-time, since it has extremely low computation complexity with simple linear and nonlinear transform units. Phase-only synthesis based on back-propagation neural network was first implemented in [14], in which the scanning range is divided into several sectors and then the main beam and null directions are being coded according to the sectors act as the network inputs. The efficiency of the network is demonstrated, however, it cannot control the nulls and desired direction of the array pattern accurately. The interference direction was used as the input of the back-propagation neural

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Zia Ur Rahman.

network in [15], which means that different networks need to be trained for different number of interferences. Practically, the number of interferences is not known in advance. A three layer radial basis function neural network has been used to find the optimum weight vector for a robust and fast beamformer [16], the covariance matrix of the array is used as the input of the network, the approach shows nearly optimal weight vector of the beamformer with small dataset. Since the performance of these neural network based methods heavily relied on the generalization ability of the network, it is essential to establish a large enough dataset to cover the distributions of different signal number and signal directions. With the limited hidden neurons, the generalization ability of radial basis function neural network for the complex array synthesis problems is restricted. Methods based on deep learning are able to reconstruct the complicated models [17], [18], which has been successfully applied in the direction of arrival of the antenna arrays with a large dataset [19]. With a high model capacity, the deep neural network (DNN) can learn a nonlinear function from a large quantities of training data and generalize well. However, it is hard to generate a large enough dataset for the phase-only array nulling DNN model because it takes long time in the calculation of the phase-only array nulling. Thus, it is not only critical to emulate the complicated array nulling model based on DNN model but also necessary to find an efficient way to prepare the array nulling dataset.

Besides, the optimum design of the antenna array is difficult to accomplish because of the array imperfections, such as the amplitude and phase errors, the effect of mutual coupling, etc [20], [21]. All these uncertainties of the array would result in the increase of the null and sidelobe levels, what's more, it will shift the nulling directions. As a matter of fact, the error not only occurs during the nulling operation but also changes with the operation conditions, which means that it is hard to estimate. Generally, the error of the array can be considered as the error in the steering vector. It has been proved that robustness is a fundamental attribute for the well generalized machine learning model [22]. When using the information of the steering vector of the array as the input of the DNN, the DNN based array nulling is expected to be robust to the imperfections of the antenna array without using any prior array imperfection information.

In this paper, to realize the phase-only array nulling with sidelobe level control in real-time, a phase-only array nulling model based on DNN is proposed. Firstly, a phase-only array synthesis model with sidelobe control is constructed. For the sake of preparing a large enough dataset for the DNN in acceptable time, the model is then relaxed to a convex problem by semidefinite relaxing and solved by direct iterative rank refinement(DIRR), which converges fast in getting the approximated rank one solution of the phase-only array nulling. After that, the DNN model is established for the phase-only array synthesis model. A large set of solutions of the relaxed convex array synthesis model are used to train the DNN. The trained DNN aims to maintain the desired array pattern properties while steering nulls in the interference directions. Experiments for different number of array elements are provided to validate the effectiveness of the proposed method. Besides, the robustness of the proposed DNN array nulling model is analyzed.

The outline of the paper is as follows. Section II formulates the phase-only array nulling model and solves the model by DIRR. In section III, the DNN model is constructed and described in detail about how it fits the phase-only nulling model. The array imperfections is listed in section IV. Numerical results are provided in section V and conclusions are drawn in section VI. Throughout the paper, bold lower case and bold capital letters represent vectors and matrices; $\Re()$ and $\Im()$ denote the real and the imaginary parts of their arguments; $()^T$ and $()^H$ are the transpose operation and the Hermitian transpose operation respectively.

II. PHASE-ONLY ARRAY NULLING MODEL

Let us assume an N element linear antenna array, the radiation pattern of the array can be written as:

$$f(u) = \boldsymbol{a}(u)^T \boldsymbol{w} \tag{1}$$

where $\boldsymbol{a} = [1, e^{2j\pi du/\lambda}, \ldots, e^{2j\pi du(N-1)/\lambda}]^T$, λ is the wave length, d is the elements spacing, $u = sin\theta$, θ is the observation angle with respect to the broadside direction and $j = \sqrt{-1}$. $\boldsymbol{w} = [w_0, w_1, \ldots, w_{N-1}]^T$, $w_n = \alpha_n e^{j\phi_n}$ is the excitation weight of the nth element, combining the amplitude α_n and the phase shift ϕ_n.

A. PHASE-ONLY ARRAY NULLING MODEL CONSTRUCTION

The radiation power of the antenna array for a given direction u is

$$|f(u)|^2 = \boldsymbol{w}^H \boldsymbol{a}(u)\boldsymbol{a}(u)^H \boldsymbol{w} \tag{2}$$

The radiation power can be equivalently formulated by the real vectors:

$$|f(u)|^2 = \boldsymbol{x}^T \boldsymbol{A}(u)^T \boldsymbol{A}(u)\boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x}$$
$$with\ \boldsymbol{A} = \begin{bmatrix} \Re(\boldsymbol{a}(u)^T) & -\Im(\boldsymbol{a}(u)^T) \\ \Im(\boldsymbol{a}(u)^T) & \Re(\boldsymbol{a}(u)^T) \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \Re(\boldsymbol{w}) \\ \Im(\boldsymbol{w}) \end{bmatrix} \tag{3}$$

Since matrix $\boldsymbol{Q}$ is a real symmetric matrix and $\boldsymbol{x}$ is a real vector, assuming that $\boldsymbol{x}\boldsymbol{x}^T = \boldsymbol{X} \in \mathbb{R}^{2N\times 2N}$, the radiation power of the array becomes

$$|f(u)|^2 = \boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} = Tr(\boldsymbol{Q}\boldsymbol{X}) \tag{4}$$

where $Tr(\boldsymbol{B})$ is the trace of matrix $\boldsymbol{B}$, $\boldsymbol{X}$ should be a rank one matrix.

In the phase-only array synthesis, the excitation amplitude of the elements remain unchanged, thus the total output power of the array is fixed. The increase of sidelobe level of the array pattern will lead to the decrease of main lobe gain. To reduce the number of constraints, the sidelobe control is replaced by controlling the desired direction gain since the array output power is fixed. The phase-only array synthesis model can be expressed as:

$$\begin{aligned}
&min\ Tr(\boldsymbol{Q}_i\boldsymbol{X}), && u\in\Omega_i \\
&s.t.\ Tr(\boldsymbol{Q}_d\boldsymbol{X})\geq\delta_d, && u\in\Omega_d && (a)\\
&\quad Tr(\boldsymbol{Q}_m\boldsymbol{X})\leq Tr(\boldsymbol{Q}_d\boldsymbol{X}), && u\in\Omega_m && (b)\\
&\quad Tr(\boldsymbol{Q}_n\boldsymbol{X}) = {\alpha_n}^2, && n=0,1,\ldots,N-1 && (c)\\
&\quad rank(\boldsymbol{X}) = 1 && && (d)
\end{aligned} \tag{5}$$

where Ω_i, Ω_m, Ω_d denotes the interference directions, the region near the desired direction and the desired direction respectively. $\boldsymbol{Q}_n$ matrix composed of all zeros except $\boldsymbol{Q}_n(n, n)$ and $\boldsymbol{Q}_n(n+N, n+N)$ equal 1. The optimization is to minimize the power at the interference directions while satisfying the following constraints: preserving the desired direction power which is expressed by constraint (a), maintaining the direction stable which is described with constraint (b), forcing the array element amplitude unchanged while alternating the phase of the elements to form nulls in the array pattern which is shown in constraint (c) and remaining the rank of matrix $\boldsymbol{X}$ to one which is denoted as constraint (d).

The optimization problem in (5) is a NP-hard problem because of the rank constraint. When relaxing the rank constraint, the optimization in (5) becomes a semidefinite programming problem (SDP), which can be solved by convex optimization using interior point methods [23]. However, standard interior point methods almost never return a low-rank solution. To get low rank solutions, several iterative techniques have been proposed to minimize the rank of the solution [24]. In this paper, direct iterative rank refinement is applied to solve (5), because it converges fast to an approximately rank one solution matrix [25].

B. DIRECT ITERATIVE RANK REFINEMENT FOR THE ARRAY SYNTHESIS MODEL

Since $\boldsymbol{X}$ is a symmetric matrix, its singular value decomposition is formed as follows:

$$\boldsymbol{X} = \sum_{n=0}^{2N-1} \sigma_n \boldsymbol{v}_n \boldsymbol{v}_n^T \tag{6}$$

where $\sigma_1\geq\cdots\geq\sigma_{2N}$ are the ordered singular values and $\boldsymbol{v}_n$ is the corresponding singular vector. When the ratio R

$$R = \sigma_1/(\sum_{n=0}^{2N-1}\sigma_n) \tag{7}$$

approximates 1, the solution of $\boldsymbol{X}$ is approximated as rank one, then the candidate solution of the model can be formulated as

$$\tilde{\boldsymbol{w}} = \sqrt{\sigma_1}\boldsymbol{v}_1 \tag{8}$$

Since the solution obtained from the convex optimal is not always the rank one solution, hence, we are going to iteratively minimize the rank of $\boldsymbol{W}$. At each iteration k, the following SDP is solved:

$$\begin{aligned}
&min\ Tr((1-\gamma_k)\boldsymbol{Q}_i - \gamma_k \boldsymbol{v}_1^k {\boldsymbol{v}_1^k}^T)\boldsymbol{X}^{k+1}, && u\in\Omega_i\\
&s.t.\ Tr(\boldsymbol{Q}_d\boldsymbol{X})\geq\delta_d, && u\in\Omega_d\\
&\quad Tr(\boldsymbol{Q}_m\boldsymbol{X})\leq Tr(\boldsymbol{Q}_d\boldsymbol{X}), && u\in\Omega_m\\
&\quad Tr(\boldsymbol{Q}_n\boldsymbol{X}) = {\alpha_n}^2, && n=0,1,\ldots,N-1
\end{aligned} \tag{9}$$

where γ_k is the parameter controlling the trade off between the power of the array pattern at the interference direction and the rank of the obtained solution. Here, γ_k is chosen as a function of R in the kth iteration:

$$\gamma_k = \begin{cases} 0, & k=1\\ 1-\sigma_1^k/\sum_{n=1}^{2N}\sigma_k^n, & k>1 \end{cases} \tag{10}$$

By maximizing the energy of $\boldsymbol{X}^k$ along the singular vector, the approximate rank one solution can be obtained in a few iterations. The corresponding excitation phase of the n-th element is obtained as:

$$\phi_n = angle(\boldsymbol{v}_1(n) + j\boldsymbol{v}_1(n+N)) \tag{11}$$

where $angle()$ denotes the phase angle of the complex value.

III. PATTERN NULLING BASED ON DEEP NEURAL NETWORK

Deep neural network generally consists of several cascaded layers. Each layer in the network consists of several neurons. For a fully connected network, every neuron in every layer is connected to each neuron in the previous layer. Such a connection is mathematically defined as a linear transformation using a weight matrix with the addition of a bias, which is then followed by a non-linear activation function. Assuming that the activation function applied to the l-th layer is denoted by h_l, the relation between the $(l-1)$-th layer and the l-th layer can be expressed as

$$\boldsymbol{Z}_l = h_l(\boldsymbol{W}_l\boldsymbol{Z}_{l-1} + \boldsymbol{b}_l) \tag{12}$$

where $\boldsymbol{Z}_{l-1}$ and $\boldsymbol{Z}_l$ is the output of the $(l-1)$-th layer and the output of the l-th layer, $\boldsymbol{W}_l$ is the weight matrix of the l-th layer and $\boldsymbol{b}_l$ is a bias vector.

In the training, the network learns the behavior of the array synthesis model by adjusting the parameters of the neural network to minimize the loss function of the network. After training, the trained network can predict the phases of the array elements with given input. Because of the generalization ability, the trained DNN can predict the phases of the array elements with similar performance of the original array synthesis model even though the input of the network never appears in the trainset.

A. PROPOSED DNN MODEL

The DNN model is a regression model that aims to approximate the array nulling model formulated by (5). In this paper, we devised a deep neural network specially for the

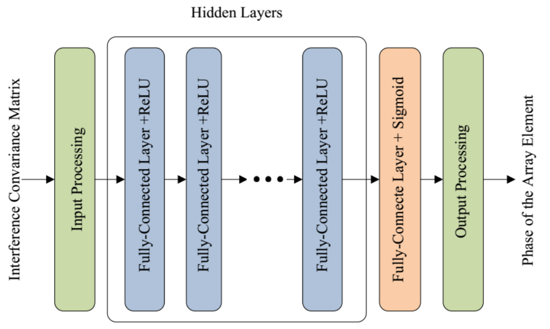

Hidden Layers
Interference Convariance Matrix
Input Processing
Fully-Connected Layer +ReLU
Fully-Connected Layer +ReLU
Fully-Connected Layer +ReLU
Fully-Connecte Layer + Sigmoid
Output Processing
Phase of the Array Element

FIGURE 1. The structure of the DNN model for the phase-only array nulling.

phase-only array nulling. The DNN structure is given in Fig.1. The activation function, the training dataset structure and the loss function of the DNN is particularly chosen to fit the array nulling model better. The network consists of the input preprocessing, several hidden layers, an output layer and output processing. We choose the ReLU function ($ReLU(z) = max(0, z)$) as the activation function for the hidden layers to escape the vanishing gradient problem. As the model is a regression learning model, sigmoid function ($Sigmoid(z) = \frac{1}{1+e^{-z}}$) is chosen as the activation function for the output layer.

B. TRAINNING DATASET STRUCTURE

Besides the model described in the previous section, the training dataset plays important roles in the performance of the DNN. For the purpose of decreasing the input neuron number and the trainset number of the DNN, the offsets between the interference directions and the desired direction are adopted as the unknown parameters of the network. Then, the covariance matrix $\boldsymbol{R}$ of the offset steering vector is considered as the input of the network:

$$\boldsymbol{R} = \sum_{i=1}^{q} \boldsymbol{a}(\tilde{u})\boldsymbol{a}(\tilde{u})^H \tag{13}$$

where q is the number of the interferences and $\tilde{u}$ is the offset between the interference direction and the desired direction.

Since the activation function of the DNN model in Fig.1 can not deal with the complex numbers, the covariance matrix of the steering vector has to be divided into two layers: the first layer is the real part of matrix $\boldsymbol{R}$, the second layer is the imaginary part of the matrix $\boldsymbol{R}$. Both the input and the output of the network are normalized between [0, 1]. The output of the network is the normalized phase of the antenna element which will convert to excitation phase after output post-processing.

C. LOSS FUNCTION OF THE DNN MODEL

The network processes the input data by propagating through its layers and adjusting the weight and bias of the neurons to minimize the difference between the output of the trainset and DNN. The difference is measured by the loss function of the DNN. Since the output of the phase-only convex model is the approximated rank one solution, which is not always the optimal phase of the antenna array, Huber Loss function is chosen as the loss function of the network. The Huber Loss is widely used in robust regression because it is less sensitive to outliers than the squared error loss [26]. It comprises two parts, corresponding to the l_1 and l_2 losses. Assuming that the error between the network output and the corresponding actual output value is t, the loss function can be formally defined as:

$$H_e(t) = \begin{cases} \frac{t^2}{2} & \text{if } |t| < c \\ ct - \frac{c^2}{2} & \text{if } |t| \geq c \end{cases} \tag{14}$$

where c is the cutting edge parameter. With the same input, the output of the l_1 penalty is smaller than the output of the l_2 penalty function. l_1 penalty function has to be applied when the absolute of the error exceeds the cutting edge. Thus, even the output of the phase-only convex model is the approximated solution, the network can generalize well.

IV. CALCULATION OF THE ARRAY IMPERFECTIONS

The array imperfections typically contain the elements position error, the excitation error of the elements and the mutual coupling between the elements. All these imperfections can be simplified as the inconsistence of the steering vectors [21]:

$$\boldsymbol{a}_e(u) = \Gamma \boldsymbol{a}(u) \tag{15}$$

where $\Gamma = \boldsymbol{I} + diag(\varepsilon[r_0, r_1, \ldots, r_{N-1}])$, $\boldsymbol{I}$ is the identity matrix, r_n are complex random variables with $|r_n| \leq 1$, the array error tolerance is ε.

When adopts the steering vector with the array imperfections, the correction of the offset steering vector $\boldsymbol{R}$ can be expressed as:

$$\boldsymbol{R}_e = \sum_{i=1}^{q} \boldsymbol{a}_e(\tilde{u})\boldsymbol{a}_e(\tilde{u})^H \tag{16}$$

$\boldsymbol{R}_e$ is applied to validate the robustness of the trained DNN model. Because the DNN model does not utilize the prior information of the array imperfections. Although the robustness is measured by the simplified formulation of error imperfection, it is reasonable to verify the robust performance.

V. SIMULATION RESULTS

To verify the performance of the DNN based array nulling, the simulation has been carried out on the 10-element and 16-element linear arrays.

In the relaxed convex phase-only nulling model, the desired direction distortion is set as less than 0.25dB and the initial amplitude of the array has $-15dB$ Taylor taper. The simulation environment is based on Python 3.6.3 with TensorFlow 1.2.1 and Keras 1.0.6 on a laptop with 4 Intel i7-6500 CPU Cores 2.5GHz, and 8GB of memory. The same initialization is set for the two DNN model. The DNN updates

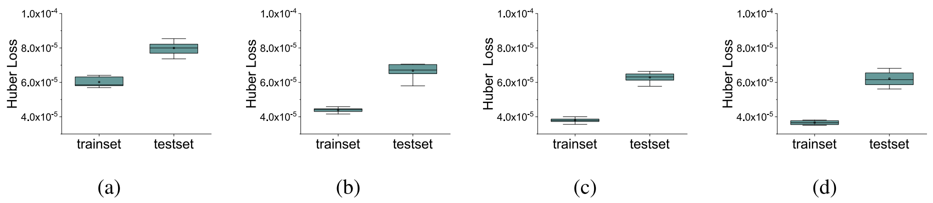


FIGURE 2. Huber Loss of the 10-element array nulling DNN model with different number of hidden layers: (a) three hidden layers with 128, 256 and 128 neurons; (b) four hidden layers with 128, 256, 256 and 128 neurons; (c) five hidden layers with 128, 256, 256, 256 and 128 neurons; (d) six hidden layers with 128, 256, 256, 256, 256 and 128 neurons.

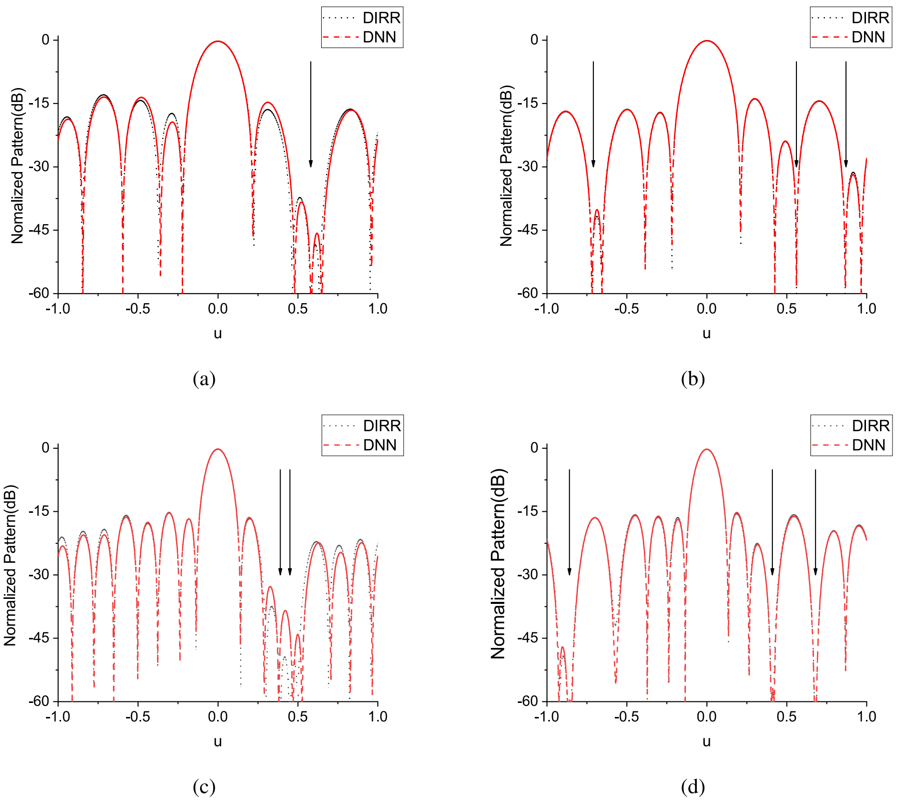


FIGURE 3. Radiation pattern comparison between the DIRR solution and DNN for 10-element and 16-element linear array: (a) optimized 10-element array pattern with one null at 0.58; (b) optimized 10-element array pattern with three nulls at −0.71, 0.56 and 0.87; (c) optimized 16-element array pattern with two nulls at 0.39 and 0.45; (d) optimized 16-element array pattern with three nulls at −0.86, 0.41 and 0.68.

the weights and biases of the neurons via the adaptive moment estimation. The cutting edge of the Huber Loss function is set as 0.03. The initial value of the learning rate of the DNN is 0.001. During the training, the learning rate is decreased adaptively according to the value of Huber Loss of the proposed DNN model. 20000 input-output pairs of data is used to train the network within 100 epochs. The interference angle set is generated from the sidelobe regions by the random sampling. The inputs and the outputs of the DNN model trainset are constructed according to the interference angle sets respectively. In the training, the inputs are generated from the ideal steering vectors of the interference angles. Besides, the maximum number of the interference of the array in the simulation is 3.

A. EVALUATION OF THE DNN MODEL

The Huber Loss generated from the 10-fold cross validation of the 10-element array synthesis DNN models with different number of hidden layers are given in Fig.2. The Huber Loss of the DNN decreases slower when the hidden layers are greater than 4. Thus, in the following evaluation of the array synthesis based on the DNN model, four hidden layer DNN models are constructed for the 10-element array and 16 element array. The mean DNN Huber Losses from the 10-fold cross validation are 4.4×10^{-5} in the trainset and 6.6×10^{-5} in the testset for the DNN model based on 10-element array synthesis and 4.6×10^{-5} in the trainset and 7.2×10^{-5} in the testset for the DNN model based on 16-element array synthesis respectively.

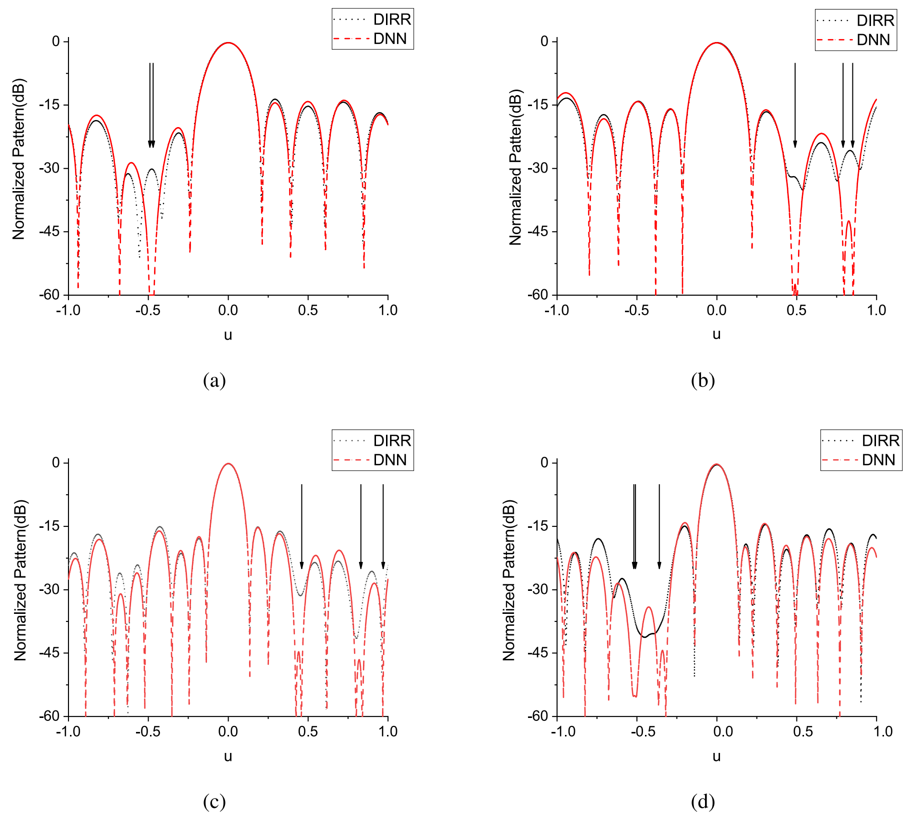


FIGURE 4. Radiation pattern comparison between the DIRR solution and DNN for 10 element and 16 element linear array when the error tolerance is 0.1: (a) optimized 10-element array pattern under array imperfection with two nulls at −0.49 and −0.47; (b) optimized 10-element array pattern under array imperfection with three nulls at 0.49, 0.79 and 0.85; (c) optimized 16-element array pattern under array imperfection with three nulls at 0.46, 0.83 and 0.97; (d) optimized 16-element array pattern under array imperfection with three nulls at −0.52, −0.51 and −0.36.

TABLE 1. Performance comparison of DIRR and DNN for different element number.

Scenarios	10 element		16 element	
	DIRR	DNN	DIRR	DNN
DDL_{mean}	$-0.20dB$	$-0.20dB$	$-0.19dB$	$-0.18dB$
SLL_{mean}	$-12.76dB$	$-12.75dB$	$-14.70dB$	$-14.71dB$
NL_{mean}	$-43.59dB$	$-42.07dB$	$-51.13dB$	$-46.46dB$

Both the DNN models for the 10-element array and the 16-element array with four hidden layer are further tested with 2000 data which never appears in the training set. The test data is also selected randomly from the interference angle set. The performance of the DNN model is compared with the DIRR solutions of the convex phase-only nulling model. Some examples of the array pattern generated by the DNN model are shown in Fig. 3, which shows that both the DNN model achieves the desired array patterns with notching in the interference direction and preserving the desired direction gain.

The mean desired direction loss(DDL_{mean}), mean sidelobe level(SLL_{mean}) and mean null level(NL_{mean}) of the array pattern are used to measure the performance of the DNN model. As shown in Table 1, the network works well in exhibiting similar performance with the original phase-only nulling model. Both the DNN models succeed in forming nulls at the interferences with almost the same behavior as the original phase-only nulling model in the desired direction loss and the sidelobe level property.

It takes less than 5 iterations for DIRR to get the approximate rank one solution of the phase-only array synthesis model, which consumes 0.3s and 0.8s for convex optimization in solving equation (9) for the 10 element and 16 element array respectively. However, it takes less than 10^{-3} seconds with DNN to get the corresponding array element phase. These two properties demonstrate that the DNN model succeed in realizing the given phase-only array synthesis model in real-time.

B. ROBUST ANALYSIS

Fig. 4 compares some array patterns of the DNN model and the DIRR solutions when the array error tolerance ε is 0.1. With the same imperfect covariance matrix, the null level of the array pattern based on the original array synthesis deteriorates, while the DNN model still forms nulls at the interference directions.

Since the array imperfections, such as the amplitude response error, can be taken as the Gaussian random distributed. In order to further verify the robustness of the network, we do Monte Carlo analysis on the ratio between the signal direction power and the interference direction power (SINR) under different situations in this section. For

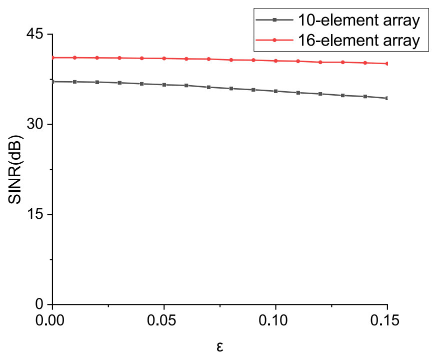


FIGURE 5. Output SINR with different array error tolerance.

all simulations, Monte Carlo analysis is carried out with 50 times. Then, the mean SINR of the 2000 test data is used to validate the robustness of the network.

As shown in Fig. 5, with the increase of the array imperfection parameter ε, the SINR of the DNN model decreases slowly, which demonstrates that the DNN model adapts well to the array uncertainties.

VI. CONCLUSION

In this paper, a DNN model is proposed to emulate the phase-only array nulling. The on-line phase-only array synthesis based on the DNN is more efficient in terms of computation time. Simulation results on the 10-element linear array and 16-element linear array show that the trained DNN model exhibit similar performance with the constructed phase-only nulling model, which could form nulls in the interference directions and maintain the gain of the desired direction. Despite the capability of preserving the desired array pattern properties, the DNN model also adapts well to the array imperfections. All these properties make the array synthesis based on DNN very useful in practice.

However, it takes a long time to prepare a large amount of data and train the DNN off-line for the array synthesis model. Therefore, the potential further work is to optimize the neural network parameters and implement the training process to reduce the number of the dataset under different array synthesis requirements.

REFERENCES

[1] R. L. Haupt, *Antenna Arrays: A Computational Approach*. Hoboken, NJ, USA: Wiley, 2010.

[2] H. Lebret and S. Boyd, "Antenna array pattern synthesis via convex optimization," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 526–532, Mar. 1997.

[3] X. Zhang, Z. He, B. Liao, X. Zhang, Z. Cheng, and Y. Lu, "A^{2}RC: An accurate array response control algorithm for pattern synthesis," *IEEE Trans. Signal Process.*, vol. 65, no. 7, pp. 1810–1824, Apr. 2017.

[4] G. Bellizzi and O. M. Bucci, "On the optimal synthesis of sum or difference patterns of centrosymmetric arrays under arbitrary sidelobe constraints," *IEEE Trans. Antennas Propag.*, vol. 66, no. 9, pp. 4620–4626, Jan. 2018.

[5] H. Steyskal, "Simple method for pattern nulling by phase perturbation," *IEEE Trans. Antennas Propag.*, vol. AP-31, no. 1, pp. 163–166, Jan. 1983.

[6] S. T. Smith, "Optimum phase-only adaptive nulling," *IEEE Trans. Signal Process.*, vol. 47, no. 7, pp. 1835–1843, Jul. 1999.

[7] R. L. Haupt, "Phase-only adaptive nulling with a genetic algorithm," *IEEE Trans. Antennas Propag.*, vol. 45, no. 6, pp. 1009–1015, Jun. 1997.

[8] A. Chatterjee and G. K. Mahanti, "Combination of fast Fourier transform and self-adaptive differential evolution algorithm for synthesis of phase-only reconfigurable rectangular array antenna," *Ann. Telecommun.*, vol. 69, pp. 515–527, Oct. 2013.

[9] T. Van Luyen and T. V. B. Giang, "Interference suppression of ULA antennas by phase-only control using bat algorithm," *IEEE Antennas Wireless Propag. Lett.*, vol. 16, pp. 3038–3042, 2017.

[10] P. J. Kajenski, "Phase only antenna pattern notching via a semidefinite programming relaxation," *IEEE Trans. Antennas Propag.*, vol. 60, no. 5, pp. 2562–2565, May 2012.

[11] B. Fuchs, "Application of convex relaxation to array synthesis problems," *IEEE Trans. Antennas Propag.*, vol. 62, no. 2, pp. 634–640, Feb. 2014.

[12] A. Rawat, R. N. Yadav, and S. C. Shrivastava, "Neural network applications in smart antenna arrays: A review," *AEU-Int. J. Electron. Commun.*, vol. 66, no. 11, pp. 903–912, 2012.

[13] A. Massa, G. Oliveri, M. Salucci, N. Anselmi, and P. Rocca, "Learning-by-examples techniques as applied to electromagnetics," *J. Electromagn. Waves Appl.*, vol. 32, no. 4, pp. 516–541, 2018.

[14] R. Ghayoula, N. Fadlallah, A. Gharsallah, and M. Rammal, "Phase-only adaptive nulling with neural networks for antenna array synthesis," *IET Microw. Antennas Propag.*, vol. 3, no. 1, pp. 154–163, 2009.

[15] Z. D. Zaharis, C. Skeberis, T. D. Xenos, P. I. Lazaridis, and J. Cosmas, "Design of a novel antenna array beamformer using neural networks trained by modified adaptive dispersion invasive weed optimization based data," *IEEE Trans. Broadcast.*, vol. 59, no. 3, pp. 455–460, Sep. 2013.

[16] T. Sallam, A. B. Abdel-Rahman, M. Alghoniemy, Z. Kawasaki, and T. Ushio, "A neural-network-based beamformer for phased array weather radar," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 9, pp. 5095–5104, Sep. 2016.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016.

[18] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, Mar. 2017.

[19] Z.-M. Liu, C. Zhang, and P. S. Yu, "Direction-of-arrival estimation based on deep neural networks with robustness to array imperfections," *IEEE Trans. Antennas Propag.*, vol. 66, no. 12, pp. 7315–7327, Dec. 2018.

[20] C. M. Schmid, S. Schuster, R. Feger, and A. Stelzer, "On the effects of calibration errors and mutual coupling on the beam pattern of an antenna array," *IEEE Trans. Antennas Propag.*, vol. 61, no. 8, pp. 4063–4072, Aug. 2013.

[21] Y. Xu, X. Shi, A. Wang, and J. Xu, "Design of sum and difference patterns with common nulls and low SLLs simultaneously in the presence of array errors," *IEEE Trans. Antennas Propag.*, vol. 67, no. 2, pp. 934–944, Feb. 2019.

[22] H. Xu and S. Mannor, "Robustness and generalization," *Mach. Learn.*, vol. 86, no. 3, pp. 391–423, 2012.

[23] (2012). *CVX: MATLAB Software for Disciplined Convex Programming, Version 2.0 Beta CVX Research*. [Online]. Available: http://cvxr.com/cvx

[24] M. Fazel, H. Hind, and S. Boyd, "Rank minimization and applications in system theory," in *Proc. Amer. Control Conf.*, Boston, MA, USA, Jun. 2004, pp. 3273–3278.

[25] M. Dedeoğlu, Y. K. Alp, and O. Arıkan, "FIR filter design by convex optimization using directed iterative rank refinement algorithm," *IEEE Trans. Signal Process.*, vol. 64, no. 9, pp. 2209–2219, May 2016.

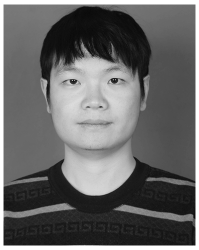
[26] C. Xi, F. Wang, V. K. Devabhaktuni, and Q.-J. Zhang, "Huber optimization of neural networks: A robust training method [microwave modeling]," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Washington, DC, USA, Jul. 1999, pp. 1639–1642.

ZHONGHUI ZHAO was born in Shandong, China, in 1991. She received the B.S. degree in communication engineering from Northwest University, Xi'an, China, in 2009, and the M.S. degree in electromagnetic field and microwave technology from Northwestern Polytechnical University, Xi'an, Shaanxi, in 2016, where she is currently pursuing the Ph.D. degree with the School of Electronics and Information.

Her current research interests include antenna array signal processing and machine learning.

HUILING ZHAO was born in Shaanxi, China, in 1967. She received the Ph.D. degree in circuit and system from Northwestern Polytechnical University, Xian, China, in 2002.

She was a Visiting Associate Professor with the Duke University, USA, from 2006 to 2007. She currently serves as a Professor with the School of Electronics and Information, Northwestern Polytechnical University. Her current research interests include optimization algorithm development for beamforming, computational electromagnetic, and antenna design.



MINGXUAN ZHENG was born in Shaanxi, China, in 1990. He received the B.S. and M.S. degrees in electromagnetic field and wireless technology from Northwestern Polytechnical University, Xi'an, China, in 2008 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronics and Information.

His current research interests include antenna array and computational electromagnetic.

JUNJIE TANG was born in Chongqing, China, in 1994. He received the B.S. degree in electromagnetic field and wireless technology from Northwestern Polytechnical University, Xi'an, China, in 2017, where he is currently pursuing the graduate degree in electromagnetic field and microwave technology.

His current research interests include antenna array and electromagnetic absorber.