A Novel Realistic Approach of Adaptive Beamforming based on Deep Neural Networks

Ioannis Mallioras, *Student Member, IEEE*, Zaharias D. Zaharis, *Senior Member, IEEE*, Pavlos I. Lazaridis, *Senior Member, IEEE*, and Stelios Pantelopoulos

***Abstract*—A new deep neural network (NN) approach applied to antenna array adaptive beamforming is presented in this paper. A recurrent NN (RNN) based on the gated recurrent unit (GRU) architecture is used as a beamformer in order to produce proper complex weights for the feeding of the antenna array. The proposed RNN utilizes four hidden GRU layers and one extra layer for linear transformation. The produced weights are subsequently compared with respective weights derived by a null steering beamforming (NSB) technique in order to measure the accuracy of the RNN. The RNN training is performed by using a large data set derived from an NSB technique applied to a realistic microstrip linear antenna array, in order to take into account real-world effects, like the non-isotropic radiation pattern of an array element and the mutual coupling between the array elements. The RNN performance is examined by using the root mean square error metric, whereas its beamforming performance is evaluated by estimating the mean value and the standard deviation of the divergences of the main lobe and nulls directions from their respective desired directions. A comparison between various NN structures and an overall study of the proposed RNN-based beamformer are also presented.**

***Keywords*—Adaptive beamforming, antenna beamforming, deep learning, neural networks, recurrent neural networks, smart antennas.**

I. Introduction

THE demand of the modern wireless communications landscape for faster and more reliable networks is constantly increasing. Smart antennas can potentially be a solution to these demands by employing self-regulating algorithms to control the transmission and reception of signals. From estimating the direction of arrival (DoA) of incoming signals, to producing the desired radiation pattern, and to finally establishing low level communication noise, intelligent algorithms have become an effective solution because they significantly decrease the latency of the beamforming process. In this way, an antenna array can dynamically steer the main lobe of its radiation pattern towards the direction of a desired incoming signal (i.e., signal of interest or SoI), while placing nulls towards the directions of respective interfering signals (i.e., signals of avoidance or SoAs), to finally maximize the signal to interference-plus-noise ratio (SINR). This process must be repeated every time a change occurs either in DoA of SoI or in DoA of any SoA, and is called "adaptive beamforming" (ABF). ABF is the principal real-time process performed by smart antennas. It plays a vital role in ensuring the quality and stability of wireless communications in an ever-changing environment.

An antenna array consists of many small radiating elements, which work together as a single antenna. The radiation of each element is added to form the total radiation pattern, which determines the direction of the main lobe, the side lobes, and the nulls. The array elements are often non-isotropic, meaning that they do not always behave as ideal signal sources because they are influenced by their individual radiation patterns (due to their geometry) and mutual coupling phenomena. To force an element radiate, a feeding weight must be applied to it. When the antenna operates as a transmitter, these weights represent either input currents or input voltages (including amplitudes and phases), which are applied to the elements by using a proper active electronic circuit driven by the processing unit, which has calculated the appropriate weights [1]. By controlling the amplitude and phase of the feeding weight of each element, we can control the radiation pattern of the antenna array. This allows the main lobe to be steered towards the desired direction while placing nulls to the directions of interfering signals.

A lot of beamforming applications can be found in the literature [2]-[10]. A four-arm spiral antenna with a monolithic integration of a modified Butler matrix beamforming network is implemented in [2] for operation from 50 to 75 GHz. A minimum variance beamforming method with linear constraints is presented in [3] to improve the calibration efficiency of an array-fed reflector antenna. In [4], various beamforming strategies are applied to a quad-mode antenna utilizing four available excitation modes to maximize the gain, the signal to noise ratio (SNR), and the polarization discrimination, whilst retaining minimum noise over the field of view. New digital beamforming techniques aiming at improving the performance of microwave radiometers used in ocean observation missions

Manuscript received January 9, 2022. This research was supported by the European Union, through the Horizon 2020 Marie Skłodowska-Curie Innovative Training Networks Programme "Mobility and Training for beyond 5G Ecosystems (MOTOR5G)" under grant agreement no. 861219.. (*Corresponding author: Ioannis Mallioras*)

I. Mallioras and S. Pantelopoulos are with the Maggioli S.p.A., 47822 Santarcangelo di Romagna, Italy (e-mail: mallioras@auth.gr; stelios.pantelopoulos@maggioli.it).

Z. D. Zaharis is with the School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece (e-mail: zaharis@auth.gr).

P. I. Lazaridis is with the Department of Engineering and Technology, University of Huddersfield, Huddersfield HD1 3DH, U.K. (e-mail: p.lazaridis@hud.ac.uk).

are presented in [5]. A miniaturized reconfigurable antenna, operating from 3.5 to 5.5 GHz, is proposed in [6] to perform beam-steering, by properly controlling eight PIN diodes.

Furthermore, some studies demonstrate the ability of smaller antennas to easily manipulate their radiation. This ability allows beamforming to be applied to smaller devices [11]-[13]. An effective implementation in [11] shows that the mutual coupling problem can be avoided in compact antennas using spiral resonators. In [12], a compact antenna array operating in the 2.45 GHz band achieves reasonable gain values and high directivity despite its small size. In [13], a novel implementation using metamaterials significantly improves the directive radiation of a dipole antenna.

So far, several deterministic algorithms have been used as ABF techniques [14]-[21]. One of these is the null steering beamforming (NSB) technique, which produces feeding weights to steer the main lobe and place nulls with high accuracy. Despite their near-optimal performance, deterministic ABF algorithms suffer from a downgraded temporal response, due to high-complexity calculations required for the extraction of the feeding weights. Moreover, if such an algorithm operates in an iterative manner, then its temporal response becomes a major issue [22]-[24]. Therefore, the future of wireless networks relies on the implementation of ABF techniques that provide not only accuracy, but also instant response.

The fields of machine learning and deep learning have made enormous leaps in the past two decades providing solutions to problems in a wide range of scientific fields [25]. Neural networks (NNs) are known to be able to mimic high-complexity functions by implementing simple calculations, such as addition, multiplication, division, and some non-linear thresholding operations. Their fast temporal response makes them not only attractive in the field of signal processing, but also a good alternative for DoA estimation and ABF (see Fig. 1). In addition, their outputs depend exclusively on the input values, so they do not rely on the temporal stability of other antenna characteristics. Despite that the training process of NNs is time-consuming, the great advantage of NNs as beamformers lies in the fact that their training may be either an offline process or a process that runs as a concurrent thread during their actual operation, and therefore the training phase neither affects the performance nor delays the actual NN operation. To avoid degradation either in performance or in temporal response in a fast-changing environment, the processing units of the NNs can be updated continuously, based on new data, by applying training as a parallel thread during the actual NN operation, as previously mentioned.

NNs have not yet demonstrated optimal accuracy when it comes to main lobe and nulls placement. In this paper, we attempt to overcome this issue by using particular deep NN structures and by training NNs using a large dataset produced by very precise ABF techniques, such as the NSB algorithm. To further improve the quality of the training samples, we have filtered out samples of low accuracy and low SINR. More details are mentioned in section V.

This research focuses on the use of NNs as a low-complexity beamforming technique and compares different structures of deep NNs in terms of accuracy and temporal response, while proposing a new beamformer implementation based on deep recurrent neural networks (RNNs), which are built using the gated recurrent unit (GRU). The NNs presented here are trained by using large datasets produced by an NSB algorithm, which has properly been modified for realistic antenna arrays, as presented in [18]. In sections VI and VII, we demonstrate the process behind the feedforward neural network (FFNN) and RNN implementations, we show the procedure of finding the best architecture for each NN type, and we finally train and test each NN type. In section VIII, we compare all the derived NN models together with the NSB technique in terms of accuracy and temporal response, to find the most promising NN model. Section IX examines the performance of the chosen NN model as a beamformer in comparison to the NSB technique for various numbers of incoming signals. Finally, the conclusions are presented in section X.

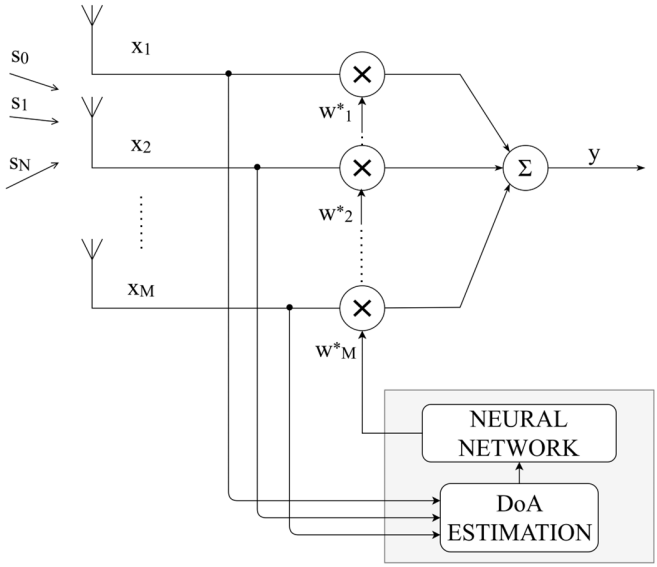


Fig. 1. Adaptive beamforming implemented by using a neural network.

II. Prior Art on NN-Based Beamforming

Already, a lot of similar studies have been conducted on the subject [14], [16], [17], [26]-[35], while the use of NNs in beamforming and other applications of smart antennas is constantly increasing [36]. Most of NNs have been trained using various ABF techniques, such as the Minimum Variance Distortionless Response (MVDR) algorithm [16], [29], [35], an Invasive Weed Optimization (IWO) variant [28], and the Least Mean Square (LMS) method [37]. When compared in [18], the MVDR and NSB techniques demonstrate similar performance, with both having an excellent ability to place nulls extremely close to DoAs of respective SoAs. We have chosen to use the NSB algorithm here, because it requires only the knowledge of angles of arrival (AoAs) of incoming signals, while the MVDR technique additionally demands the autocorrelation matrix of the signals induced at the inputs of the array elements. Some researchers have just used the autocorrelation matrix as input for their NNs [14], [17], [26], [31], thus implementing a “blind beamforming” approach, where a NN performs beamforming

without being aware of the directions of the signals. As explained in [27], the autocorrelation matrix has a highly nonlinear relation with the element weights and thus it makes training more difficult, so in this paper we have preferred to directly use AoAs of incoming signals, as in [26] and [28].

An interesting single neuron NN for a dipole antenna array is presented in [37] portraying the simplest application of NNs as beamformers, thus proving their efficiency and accuracy. In [17], a three-layer radial basis function FFNN is used to provide robustness against uncertainty in the signal AoA, thus improving the SINR. Another shallow FFNN is utilized in [31] to steer the radiation power toward the directions of the desired users and suppress interfering sources. The presented radiation patterns exhibit great main lobe accuracy and side lobe suppression, but without any information about the accuracy of the interference directions A convolutional NN (CNN) approach with three types of hidden layers is also studied in [33] to improve the SINR values at the beamformer output and suppress both narrowband and wideband interferences. Unfortunately, the model's statistical accuracy in finding the proper directions of the main lobe and the nulls is not presented numerically for comparison. Some studies focus solely on the main lobe steering without placing nulls at the exact directions of interference signals [35] or by simply nulling a wider area of the radiation pattern [30], [32]. In [28], a FFNN implementation is studied on low noise level conditions in comparison to the IWO technique, thus demonstrating the efficiency and capabilities of the NN as a beamformer. A CNN-inspired model [29] shows better performance than a conventional FFNN with a mean divergence of the main lobe from the desired direction of less than 0.6 degrees. Applications of RNNs in the field of beamforming have already shown improvements in the behaviour of millimeter wave beamforming transmitters [34]. In [35], a long short-term memory (LSTM) and a nonlinear autoregressive (NAR) NN are implemented for signal prediction, thus showing the great potential of recurrent networks as beamformers. In [30], a RNN is used to steer the main lobe towards DoA of SoI, while suppressing the sidelobe level. This implementation considers an antenna array composed of ideal isotropic elements, which are not realistic as previously explained. Also, the beamformer's accuracy is measured in terms of the main lobe direction and side lobe level, but not in terms of the nulls directions.

Judging from all this research, the NN-based beamforming is very promising and justifies further study. However, previous NN implementations either focused exclusively on the main lobe steering or created unsatisfactory main lobe and nulls divergences from the respective desired directions. Also, several NN implementations considered ideal antenna arrays and not realistic ones. Of course, an important reason for their inefficiencies was evidently the lack of training samples. The present study does not directly make comparisons with the ones previously mentioned, except for the case of [29] where divergences from the desired DoAs were measured. The proposed model provides a satisfactory answer to the real-world ABF problem by using of a GRU-RNN architecture that offers great accuracy both in the main lobe direction and in the placement of nulls. For this purpose, a sufficient training set was produced by a realistic antenna array ABF algorithm, which gave our NNs the diversity and abundance of data they needed to reach higher levels of accuracy.

III. Adaptive Beamforming Fundamentals

Let us consider an array composed of M elements, which are located in space at positions defined by respective position vectors $\vec{r}_m$, $m = 1, \dots, M$. The array receives N+1 monochromatic signals (where N<M) at wavelength λ. Each one of the incoming signals $s_n(k)$ (where k is the time sample and $n = 0, 1, \dots, N$) reach the antenna array from a DoA defined by a respective direction unit vector $\hat{u}_n$. We identify $s_0(k)$ as SoI and the rest of $s_n(k), n = 1, \dots, N$, as SoAs. It is considered that all DoAs have been derived by a DoA estimation algorithm, and then they are used as information to feed the beamformer. So, the beamformer is responsible for producing M proper complex weights w_m $(m = 1, \dots, M)$, which are consequently multiplied by signals $x_m(k)$ $(m = 1, \dots, M)$ induced at the inputs of the respective array elements to produce the array output $y(k)$.

As shown in [29], all input signals $x_m(k)$ $(m = 1, \dots, M)$ can be represented as a column vector $\mathbf{x}(k)$, which can be expressed by the following equation:

$$\mathbf{x}(k) = \mathbf{a}_0 s_0(k) + \mathbf{A}_i \mathbf{s}_i(k) + \mathbf{n}(k), \tag{1}$$

where

$$\mathbf{x}(k) = [x_1(k)\ \ x_2(k)\ \ \dots\ x_M(k)]^T, \tag{2}$$

$$\mathbf{s}_i(k) = [s_1(k)\ \ s_2(k)\ \ \dots\ \ s_N(k)]^T, \tag{3}$$

$$\mathbf{A}_i = [\mathbf{a}_1\ \ \mathbf{a}_2\ \ \dots\ \ \mathbf{a}_N], \tag{4}$$

and

$$\mathbf{a}_n = \begin{bmatrix} exp(j\beta\vec{r}_1 \cdot \hat{u}_n) \\ exp(j\beta\vec{r}_2 \cdot \hat{u}_n) \\ \vdots \\ exp(j\beta\vec{r}_M \cdot \hat{u}_n) \end{bmatrix}, \qquad n = 0, 1, \dots, N \tag{5}$$

are, respectively, the input vector, the SoA vector, the array steering matrix of SoAs, and the array steering vector that corresponds to DoA defined by unit vector $\hat{u}_n$, while β is the free space wavenumber ($\beta = 2\pi/\lambda$) and superscript T indicates the transpose operation. It has to be noted that the form of $\mathbf{a}_n$ given by (5) applies to the case of an ideal array, i.e., an array composed of M isotropic point sources, with no coupling between them. In the case of a realistic array, where the array elements are not omni-directional and there is coupling between them, a modified form of $\mathbf{a}_n$ replaces (5), as shown below. Regardless of the type of the array (ideal or realistic), the output is calculated as:

$$y(k) = \sum_{m=1}^{M} w_m^* \, x_m(k). \tag{6}$$

The above equation can also be written in the form

$$y(k) = \mathbf{w}^H \mathbf{x}(k), \tag{7}$$

where

$$\mathbf{w} = [w_1\ \ w_2\ \ \dots\ \ w_M]^T \tag{8}$$

is the excitation weight vector, while superscript *H* indicates the Hermitian transpose operation. The autocorrelation matrix of the input signals, which was mentioned in section II, is calculated as:

$$\mathbf{R}_{xx} = E[\mathbf{x}(k)\mathbf{x}^H(k)], \tag{9}$$

where *E*[.] denotes the mean value. This definition also applies to any type of array, either ideal or realistic.

IV. Null Steering Beamforming

A. Theoretical model applied to ideal antenna arrays

The theoretical NSB algorithm considers an ideal array, i.e., an array composed of *M* isotropic point sources, with no coupling between them. The array is considered here to be a linear one, so the sources are considered to be placed along the z-axis. As also presented in [18], DoAs of incoming signals s_n $(n = 0, 1, \dots, N)$ can be identified by respective AoAs θ_n $(n = 0, 1, \dots, N)$, which are defined as angles between DoAs of these signals and the z-axis (i.e., polar angles). If $I_m, m = 1, \dots, M$, are the currents used to feed the sources (array elements), then the radiation pattern is expressed by the array factor as follows:

$$AF(\theta) = \sum_{m=1}^{M} I_m e^{j\beta z_m cos\theta}, \tag{10}$$

where $z_m, m = 1, \dots, M$, are the positions of the point sources along the z-axis. When the array is in reception mode, the currents I_m become multipliers of the input signals x_m $(m = 1, \dots, M)$. We may consider that $I_m = w_m^*$, where w_m is a weight that expresses the conjugate value of the *m*th current. Then, we can write (10) as:

$$AF(\theta) = \sum_{m=1}^{M} w_m^* e^{j\beta z_m cos\theta} = \mathbf{w}^H \mathbf{a}(\theta), \tag{11}$$

where

$$\mathbf{a}(\theta) = \left[e^{j\beta z_1 cos\theta} \quad e^{j\beta z_2 cos\theta} \quad \dots \quad e^{j\beta z_M cos\theta}\right]^T \tag{12}$$

is the steering vector for any observation angle θ. The NSB algorithm is very accurate when it comes to null placement towards the interference directions, thus achieving near-zero angular divergences from these directions as shown in [18]. In addition, it serves as a precise beam-steering technique presenting negligible errors on the positioning of the main lobe. To achieve all the above, the NSB algorithm calculates the weights of the antenna elements based on the following expression:

$$\mathbf{w}_{NSB} = \mathbf{A}(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{v}_1, \tag{13}$$

where

$$\mathbf{A} = [\mathbf{a}(\theta_0) \;\; \mathbf{a}(\theta_1) \dots \mathbf{a}(\theta_N)] \tag{14}$$

and

$$\mathbf{v}_1 = [1 \;\; 0 \dots 0]^T \tag{15}$$

are, respectively, the array steering matrix of all the incoming signals $(M \times (N+1)$ matrix), and a unit vector of size *N*+1.

B. Modified model applied to realistic antenna arrays

In practice, the NSB algorithm is applied to realistic antenna arrays, i.e., arrays composed of *M* non-isotropic elements, with mutual coupling between them. Therefore, some modifications must be applied to this algorithm, to be able to calculate the proper feeding weights even in the case of a realistic array. The main change concerns the total radiation pattern, which now cannot be expressed just by the array factor.

The array considered in this paper has the same geometry with the one studied in [18]. It is a linear array composed of 16 microstrip rectangular patches ($M = 16$) designed according to the inset-feed method to easily achieve impedance matching. The patches are developed on a Rogers RT/duroid 5880 substrate and are uniformly spaced at fixed distance $d = \lambda/2$. The CST software package is used to model and optimize the array geometry under constraint that S-parameters $S_{mm} \leq -20$dB, $m = 1, \dots, 16$, at the input points of the microstrip elements. Given that the input of every element is located in the middle of the element side, which is parallel to the z-axis (see Fig. 2), the total electric theta-component E_θ produced on the xz-plane by the whole array is at least 20dB less than the value of the total electric phi-component E_φ on the same plane $(E_\theta \ll E_\varphi)$ for values of θ within the angular sector $[30°, 150°]$, as explained in [18]. Thus, the total radiation pattern within this sector on the xz-plane can be represented only by E_φ. In addition, E_φ can be expressed in the form of a linear combination, as follows:

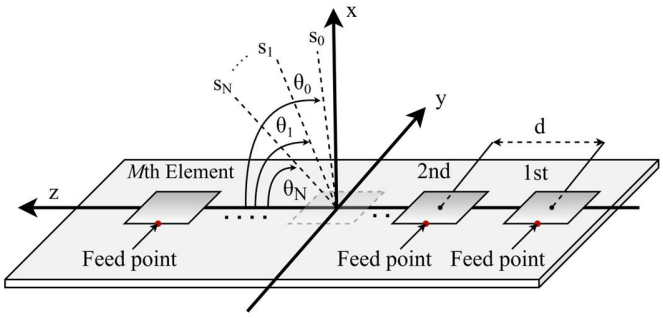


Fig. 2. Linear microstrip antenna array geometry.

$$E_\varphi(\theta) = \sum_{m=1}^{M} I_m e_{\varphi m}(\theta), \tag{16}$$

where $e_{\varphi m}(\theta)$ is the electric phi-component of the whole array, when only the *m*th element is fed by a unitary current source, while the rest of the elements are not fed by any source (parasitic elements). These components are mentioned in the literature as the "embedded element patterns" of the array, and they can be extracted by performing a full-wave analysis on the array using the CST.

By considering again that $I_m = w_m^*$, the above expression can be written as follows:

$$E_\varphi(\theta) = \sum_{m=1}^{M} w_m^* \, e_{\varphi m}(\theta) = \mathbf{w}^H \, \mathbf{e}_\varphi(\theta), \qquad (17)$$

where

$$\mathbf{e}_\varphi(\theta) = \left[e_{\varphi 1}(\theta) \;\; e_{\varphi 2}(\theta) \;\; \ldots \;\; e_{\varphi M}(\theta)\right]^T \qquad (18)$$

is a vector containing all the electric phi-components that correspond to observation angle θ. Equations (11) and (17) express the radiation patterns produced, respectively, by an ideal and a realistic antenna array. It is obvious that (11) is converted to (17) by replacing $\mathbf{a}(\theta)$ with $\mathbf{e}_\varphi(\theta)$. Since $\mathbf{a}(\theta)$ represents the steering vector of an ideal array (i.e., the theoretical steering vector), we can consider that $\mathbf{e}_\varphi(\theta)$ represents the steering vector of a realistic array, and therefore it can be called the "realistic steering vector". Furthermore, if $\mathbf{e}_\varphi(\theta)$ replaces $\mathbf{a}(\theta)$ in any equation that applies to an ideal array, then the derived equation will apply to the respective realistic array. By applying such a replacement to (14), we derive the form of the realistic steering matrix of all the incoming signals, as follows:

$$\mathbf{E}_\varphi = \left[\mathbf{e}_\varphi(\theta_0) \;\; \mathbf{e}_\varphi(\theta_1) \;\; \ldots \mathbf{e}_\varphi(\theta_N)\right] \qquad (19)$$

Finally, by replacing $\mathbf{A}$ with $\mathbf{E}_\varphi$ in (13), we derive the weight calculation formula of the NSB algorithm that applies to realistic arrays:

$$\mathbf{w}_{NSB} = \mathbf{E}_\varphi (\mathbf{E}_\varphi^H \, \mathbf{E}_\varphi)^{-1} \mathbf{v}_1 \qquad (20)$$

V. Training Dataset Production - Problem Conditions

Using (20), we are able to produce any dataset for NN training. Since the array of this study consists of 16 elements and each element is excited with a complex feeding weight, we need 32 weight numbers (i.e., 16 real and 16 imaginary parts of the complex feeding weights) to produce the radiation pattern of the array. To produce a proper dataset, some restrictions concerning the desired AoAs and their divergences from the respective actual AoAs are applied. Firstly, all the desired AoAs must lie within the angular sector $[30°, 150°]$ in order to be able to express the total radiation pattern only through E_φ ($E_\theta \ll E_\varphi$), as explained in the previous section. Secondly, all the desired AoAs have a minimum distance $\Delta\theta$, which is defined here equal to 6°. The value of $\Delta\theta$ determines the difficulty of the beamforming problem, because it is actually the minimum distance between two adjacent nulls or between the main lobe peak and its nearest null. The lower the value of $\Delta\theta$, the more difficult the problem, because the beamformer is forced to produce a radiation pattern where a null is generated at a small angular distance from another null or from the main lobe peak. Thirdly, the SNR is defined equal to 0dB, thus considering high noise conditions. Finally, we set some constraints on the accuracy of the NSB results that will be included in the training dataset. In order to make a NN produce highly accurate results, the data we use for its training must be highly accurate as well. Thus, we only keep cases where the weights derived by the NSB algorithm produce a radiation pattern that has

- main lobe divergence less than 0.5° from AoA of SoI, and
- null placement divergences less than 0.1° from AoAs of SoAs,

while we discard the rest. Correspondingly, we expect the outputs of our final NN model to perform similarly, with a small margin of error, in order to consider the NN implementation successful.

Every record of the dataset consists of:

- N+1 inputs, which are the polar angles θ_n $(n = 0, 1, \ldots, N)$, with the first one (θ_0) representing AoA of SoI and the rest of the angles representing AoAs of SoAs.
- $2M$ outputs, which are 32 weight numbers extracted by the NSB algorithm, i.e., 16 real and 16 imaginary parts of the complex weights used to feed the 16-element array.

In order to perform more effective NN training, a sufficient number of records is produced. In particular, 1.1×10^4 records are used in the process of searching the best model architecture, while 5×10^6 records are used for the training of the final model. The time needed for the production of these datasets is 42 seconds and 5.3 hours respectively (using an Intel® Core™ i7-7700HQ @2.80GHz and 8GBs of DDR4 RAM). In this way, we ensure that we will not face overfitting issues (i.e., situations where the training error is much lower than the test error) and that we create a well-generalized model.

VI. Feedforward NN Approach

In the case of FFNNs, the process of input loading and output production is simple and easy to comprehend. AoAs are loaded at the input layer of the FFNN in parallel and are consequently processed by the neurons of the hidden layers. The NN produces 32 output values, which represent the real and imaginary parts of the complex feeding weights, as shown in Fig. 3.

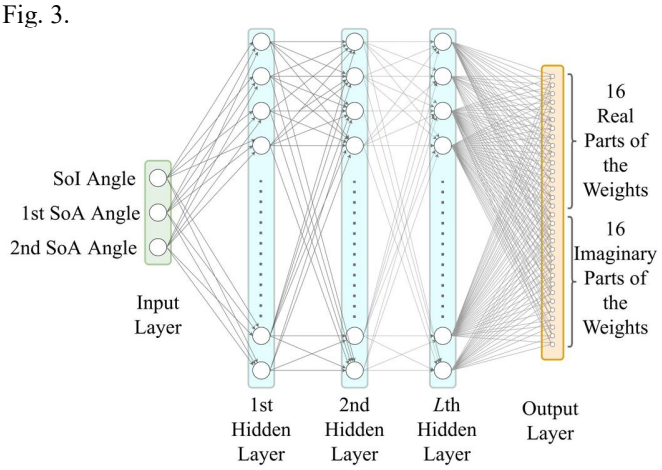


Fig. 3. L-layer FFNN implementation.

In order to find out which architecture is the best for this type of NN, we apply grid search with k-fold cross validation to figure out the best combination regarding the number of hidden layers and their respective sizes. Later on, another grid search is applied to figure out the best hyperparameter tuning concerning the batch size and the learning rate. The variations tested below refer to ABF for 3 incoming signals, i.e., 1 SoI and 2 SoAs, as shown in Fig. 3. If L is the number of hidden layers, then the first L–1 layers will be using the hyperbolic tangent (tanh) as an activation function, whereas the last layer (Lth) will be using the sigmoid activation function. To calculate the cost

for each NN weight update, we use a quadratic cost function variant, also known as root mean square error (RMSE), which is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{p}\sum_{i=1}^{p}\left(f_i - \hat{f}_i\right)^2}, \qquad (21)$$

where p is the number of neurons on the output layer of the NN ($p = 32$), $\hat{f}_i$ is the actual value of the ith output neuron, and f_i is the respective desired value of this neuron.

We also choose Adam as the optimization method and we normalize both inputs and outputs of the dataset in the range [0,1] to improve the optimization algorithm's performance and the overall training process [38]. These choices are proved to be more efficient than other alternatives after a lot of different trials. During the search process, we choose a slow learning rate, e.g., 0.001, and a batch size equal to 256, which remain the same for all our trials. Due to high complexity and non-linearity of the ABF problem, the case of one hidden layer is rejected. Instead, we test FFNN architectures with 2, 3 and 4 hidden layers. The sizes of these layers are purposely chosen to be multiples of the output size (32). Due to the size difference between input and output, it is considered good practice to select hidden layers much larger in size than the output layer, to emulate the computational complexity required to transform 3 input angles into 32 output numbers. Thus, we choose hidden layer sizes between 128 and 1024.

For this grid search, we use 10^4 records as a training set and 10^3 records as a test set. We apply a 5-fold cross validation to ensure the validity and accuracy of our results, and train each model for 500 epochs. Using this grid search, we intend to find which variation provides the lowest training and test RMSEs to choose as most promising. Each FFNN architecture is described by using the notation: [Input Layer Size, 1st Hidden Layer Size, 2nd Hidden Layer Size, …, Lth Hidden Layer Size, Output Layer Size].

The results are given in Tables I, II and III. By comparing these three tables, we notice a test RMSE drop, when we move from a two-layer FFNN to a four-layer FFNN. This is expected as an increase of a NN's depth increases its ability to model functions of high complexity [39]. Despite the increasing overfitting risk, NN models that use layers of larger size tend to achieve lower errors. This is the answer to the question why we mainly focused on using larger layer sizes, when testing a four-layer FFNN. We also observe that, while layers with size equal to 512 contribute to the models' performances, layers with size equal to 1024 simply increase the risk of overfitting. The best architecture is the one that provides the lowest training and test RMSEs, and is highlighted in bold.

Next, we look for the best hyperparameter combination by comparing different batch sizes and different learning rates. From Table IV, it seems that the best initial value for the batch size is 256 and for the learning rate is 0.001. This table also shows that testing for values of learning rate greater than 0.005 is redundant as the results become increasingly worse.

TABLE I
COMPARISON OF FFNNS WITH TWO HIDDEN LAYERS (5-FOLD CROSS VALIDATION)

Layer Sizes	Training RMSE	Test RMSE
[3, 128, 128, 32]	0.0492	0.0498
[3, 128, 256, 32]	0.0481	0.0493
[3, 256, 256, 32]	0.0483	0.0489
[3, 256, 512, 32]	0.0482	0.0488
[3, 512, 512, 32]	0.0475	0.0484

TABLE II
COMPARISON OF FFNNS WITH THREE HIDDEN LAYERS (5-FOLD CROSS VALIDATION)

Layer Sizes	Training RMSE	Test RMSE
[3, 128, 256, 256, 32]	0.0463	0.0479
[3, 128, 256, 512, 32]	0.0464	0.0477
[3, 256, 512, 512, 32]	0.0431	0.0462
[3, 256, 512, 1024, 32]	0.0448	0.0476
[3, 512, 512, 1024, 32]	0.0422	0.0459
[3, 512, 1024, 1024, 32]	0.0400	0.0456

TABLE III
COMPARISON OF FFNNS WITH FOUR HIDDEN LAYERS (5-FOLD CROSS VALIDATION)

Layer Sizes	Training RMSE	Test RMSE
[3, 128, 256, 512, 512, 32]	0.0434	0.0465
[3, 128, 256, 512, 1024, 32]	0.0455	0.0483
[3, 256, 512, 512, 1024, 32]	**0.0339**	**0.0408**

TABLE IV
BATCH SIZE COMPARISON OF FFNNS FOR VARIOUS LEARNING RATES (5-FOLD CROSS VALIDATION)

Batch Size	Learning Rate	0.001	0.005	0.010
32	Training RMSE	0.0415	0.2945	0.5707
	Test RMSE	0.0437	0.2988	0.5718
128	Training RMSE	0.0336	0.1236	0.5713
	Test RMSE	0.0415	0.1215	0.5705
256	Training RMSE	**0.0339**	0.1063	0.5000
	Test RMSE	**0.0408**	0.1043	0.4992
512	Training RMSE	0.0443	0.1990	0.4004
	Test RMSE	0.0479	0.1997	0.4012

After we have found the best settings for FFNNs applied to the ABF problem, we proceed to train and test the final model using the big dataset (5×10^6 records). The major part of this set, i.e., 4.9×10^6 records, is used for training, and the remainder, i.e., 10^5 records, is used for testing. During training, we use the Pytorch function *ReduceLROnPlateau*, which reduces the current learning rate by a factor of 0.8, if the training RMSE stops decreasing or it decreases very slowly. This learning rate regulation increases the training performance by a factor between 2 and 10, because it helps the NN training process to escape sharp local minima [40]. The results are presented in Fig. 4 and Table V. The training curve of Fig. 4 suggests that the training presents no overfitting. The sudden RMSE drops represent the times where *ReduceLROnPlateau* decided to decrease the learning rate, and it is evident that this reduction has a positive impact on the training.

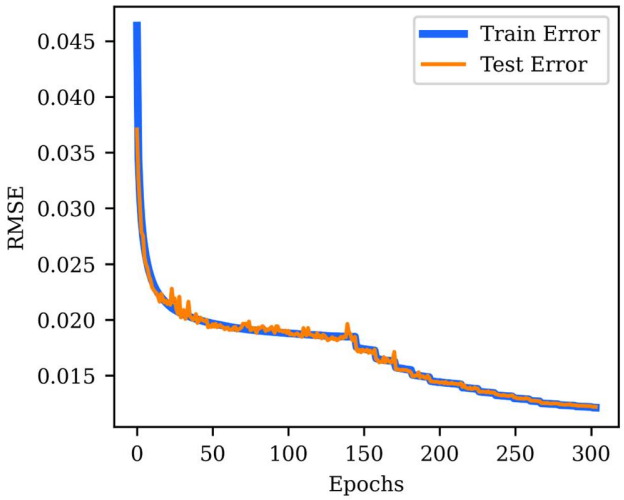


Fig. 4. Learning curves of the FFNN.

TABLE V
FFNN TRAINING AND TEST RESULTS

Epochs	Training RMSE	Test RMSE	Final learning rate	Training time (hours)
300	0.0121	0.0122	0.000055	13.28

Next, the FFNN is tested in terms of accuracy in the produced radiation patterns. The test is applied on 10^5 triads of AoAs (every triad corresponds to a SoI and two SoAs), on which the FFNN has no prior "experience". A statistical analysis of the results derived from this test is given in Table VI. It seems that the FFNN implementation is very good regarding the main lobe steering, because the mean value of the main lobe divergence derived by the FFNN satisfies the respective requirement (i.e., main lobe divergence < 0.5° as defined in section V), and it is also very close to the respective divergence derived by the NSB algorithm. However, the mean value of the nulls divergence does not satisfy the respective requirement defined in section V (i.e., nulls divergence < 0.1°), which means that the FFNN faces difficulty in placing nulls at the desired directions. This inaccuracy is also observed in the reduced mean value of the SINR by 0.67dB compared to the respective SINR value achieved by the NSB algorithm.

TABLE VI
COMPARISON BETWEEN NSB AND FFNN

	NSB [Mean/Std]	FFNN [Mean/Std]
Divergence of main lobe (°)	0.437/0.323	0.438/0.324
Divergence of nulls (°)	0.000/0.000	0.364/0.412
SINR (dB)	27.196/2.970	26.530/3.106

Finally, an example of radiation patterns produced by the NSB algorithm and the FFNN for a SoI and two SoAs is given in Fig. 5. Although the radiation patterns seem to be almost identical, the above statistical analysis shows that the FFNN is slightly inferior to the NSB algorithm in terms of nulls placement accuracy.

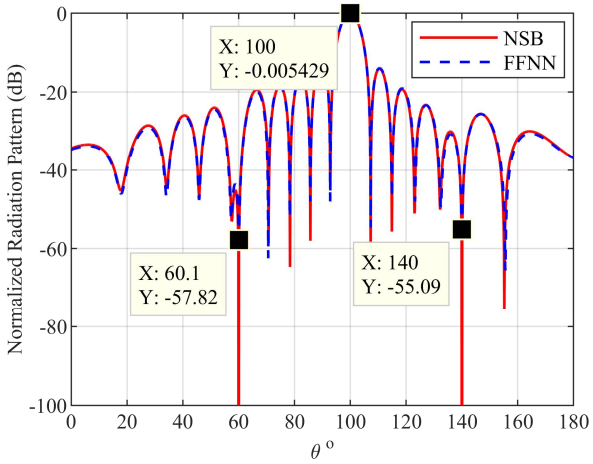


Fig. 5. Radiation patterns produced by NSB and FFNN for a SoI received at 100° and 2 SoAs received at respective AoAs equal to 60° and 140°.

VII. RECURRENT NN APPROACH

RNNs belong to another type of NNs, which are mostly used for processing sequential data. The applications of RNNs vary from speech recognition and translation to sentiment classification and music generation. They diverge from FFNNs due to their ability to exploit their "memory" to influence their outputs. Information from prior inputs can affect the outcome of the current input since data enter the RNN in sequence. RNNs can be further distinguished into categories depending on the way they produce output data. Different applications require different types of RNNs.

For the purpose of ABF, we choose the "many-to-one" approach, where AoAs enter the RNN one after another, and only the final output of the RNN is kept. When working on sentiment classification (another many-to-one approach), the input of the RNN, at each time step, is a new word not directly linked to or necessarily related to the words that came at the previous steps. However, the model is able to output a new sentiment each time we add a different word to the sentence. The new information influences the general outcome of the RNN, but at the same time all previous inputs are taken into consideration. Although these inputs do not relate to each other, the ability of the RNN to "remember" and "combine" all the inputs is what dictates the sentiment of a sentence. We have chosen this structure of NNs not because of their potential to find a relationship between the data we provide as input to the NN, but because of their ability to adapt their output depending on the new incoming information. During the prediction process of the RNN as a beamformer, the hidden states of the units travel along the different time steps. In this way, the RNN-beamformer adjusts the excitation weights progressively, in order to produce the most suitable radiation pattern for each new situation. The "progressive adaptation" of weights can be well understood in Section VIII by comparing the radiation patterns produced by the beamformer in consecutive steps.

The process followed here can also be explained by looking at the RNN structure displayed in Fig. 6. For each time step $(t = 1, 2, 3)$, the current input x_t is processed by the RNN's processing units to influence their hidden states, which are consequently passed on to the next time step's units to use. In this way, each input affects the outcome of the RNN. Once all

inputs are processed, and the output o_3 has been produced, the next input values enter the input layer to continue the training process. The basic idea is to have the weight vector initially configured by taking into account only AoA of SoI (input x_1 of Fig. 6), and then AoAs of SoAs (inputs x_2 and x_3 of Fig. 6) enter the RNN to shape the final form of the weight vector at the RNN's output.

As in the case of the FFNN, input data are fed to the RNN in batches, and the Adam optimization algorithm is used to update the hidden states, while the RMSE metric shown in (21) is used as a cost function. Given that every hidden state and consequently the output of the RNN will have a different size than 32, an extra linear transformation layer is placed after the output layer to ensure that the final outcome is a vector of 32 numbers (see Fig. 6).

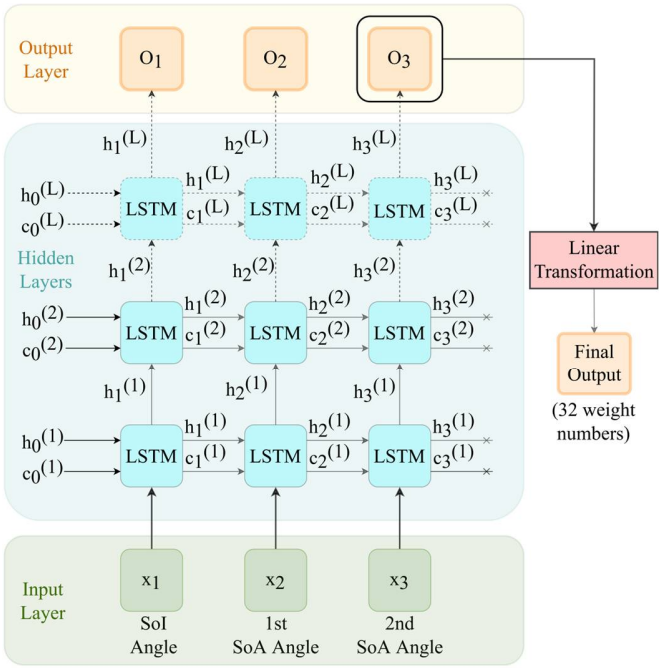
Two RNN approaches based on different processing units, i.e., LSTM and GRU, are tested. Once again, we perform grid search in both approaches for the most efficient architecture and also for hyperparameter tuning, to finally find out the most promising models. During the search process, we choose a slow learning rate, e.g., 0.001, and a batch size equal to 256, while a 5-fold and later a 3-fold cross validation are applied using, respectively, 10^4 records as a training set and 10^3 records as a test set. For the same reasons as in the FFNN case, we choose hidden layer sizes between 64 and 1024 and a number of hidden layers between 2 and 4.

In order to have a fair comparison, the preferred sizes and training parameters of the LSTM and GRU implementations must be similar.

A. *Use of LSTM units*

LSTM units are known for their ability to overcome the vanishing gradient problem of RNNs. These units employ two vectors, i.e., a "hidden state" vector, which carries information from immediately previous events, and a "cell state" vector, which provides "long term" memory capabilities as it can carry information from past events (see Fig. 6). Using a variety of gates, the LSTM units can choose which information is valuable to keep and update their hidden states and which can be tossed. The results of the grid search for 2, 3 and 4 hidden layers are presented in Table VII. It seems that the most promising architectures are those with 4 hidden layers. The lowest RMSE is observed for hidden layer sizes equal to 512 and 1024, with the latter achieving better accuracy. However, since the RMSE improvement is insignificant compared to the size increase from 512 to 1024, we decided to choose an architecture with 4 hidden layers and a hidden layer size equal to 512.

Next, we look for the best hyperparameter combination by comparing different batch sizes and different learning rates, but this time we check for learning rates between 0.001 and 0.005. The results are shown in Table VIII. It seems that the best performance is achieved when using low learning rates and low batch sizes, but in such cases the risk of overfitting increases. Additionaly, we need significantly more time for training, which makes such choices a hard compromise. By considering other alternatives that provide good performance and are less time consuming, we finally decided to choose a learning rate equal to 0.001 and a batch size equal to 256 as the best compromise between time consumption, training performance and generalization ability.



$h_i^{(j)}$: Hidden State of time step i on j-th Layer

$c_i^{(j)}$: Cell State of time step i on j-th Layer

Fig. 6. L-layer LSTM-RNN.

TABLE VII
LSTM-RNN GRID SEARCH (5-FOLD CROSS VALIDATION)

Hidden layers	Hidden Layer Size	64	128	256	512	1024
2	Training RMSE	0.0499	0.0498	0.0497	0.0491	0.0477
	Test RMSE	0.0509	0.0502	0.0500	0.0488	0.0479
3	Training RMSE	0.0501	0.0494	0.0478	0.0442	0.0446
	Test RMSE	0.0505	0.0493	0.0494	0.0456	0.0463
4	Training RMSE	0.0504	0.0494	0.0413	**0.0327**	**0.0328**
	Test RMSE	0.0502	0.0502	0.0450	**0.0402**	**0.0382**

TABLE VIII
BATCH SIZE COMPARISON OF LSTM-RNNS FOR VARIOUS LEARNING RATES (3-FOLD CROSS VALIDATION)

Batch Size	Learning Rate	0.001	0.003	0.005
32	Training RMSE	0.0181	0.0273	0.0301
	Test RMSE	0.0373	0.0395	0.0461
128	Training RMSE	0.0250	0.0319	0.0294
	Test RMSE	0.0374	0.0416	0.0430
256	Training RMSE	**0.0354**	0.0404	0.0324
	Test RMSE	**0.0406**	0.0435	0.0442
512	Training RMSE	0.0477	0.0485	0.0421
	Test RMSE	0.0496	0.0502	0.0457

After we have found the best settings for the LSTM-RNN, we proceed to train and test the final model using the big dataset (5×10^6 records). The major part of this set, i.e., 4.9×10^6 records, is used for training, and the remainder, i.e., 10^5 records, is used for testing. In comparison to the FFNN, training time is significantly increased, as shown in Table IX. The learning curves shown in Fig. 7 indicate that the training presents no

overfitting, while the sudden RMSE drops are again due to decrease in the learning rate caused by the *ReduceLROnPlateau* function. At the end of the training process after 180 epochs, the test RMSE has significantly been reduced, as shown in Fig. 7 and Table IX, thus proving the superiority of the RNNs over the FFNNs in ABF. It should be noted that the training process could continue for more than 180 epochs, but since the learning rate has already been drastically decreased and the training time is already very high, a further reduction in RMSE is considered unworthy.

Next, the LSTM-RNN is tested in terms of accuracy in the produced radiation patterns. The test is applied on 10^5 triads of AoAs (every triad corresponds to a SoI and two SoAs), on which the LSTM-RNN has no prior "experience". A statistical analysis of the results derived from this test is given in Table X. It seems that the LSTM-RNN implementation provides high accuracy regarding both the main lobe steering and the nulls placement. In particular, the mean value of the main lobe divergence derived by the LSTM-RNN satisfies the respective requirement (i.e., main lobe divergence < 0.5° as defined in section V), and it is also very close to the respective divergence derived by the NSB algorithm. In addition, the mean value of the nulls divergence is very low and satisfies the respective requirement defined in section V (i.e., nulls divergence < 0.1°). The accuracy provided by the LSTM-RNN is also verified by the mean value of the SINR, which is almost identical to the respective SINR value achieved by the NSB algorithm.

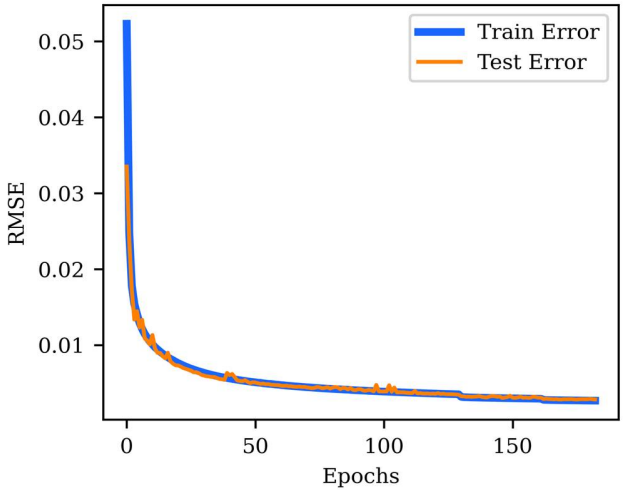


Fig. 7. Learning curves of the LSTM-RNN.

TABLE IX
LSTM-RNN TRAINING AND TEST RESULTS

Epochs	Training RMSE	Test RMSE	Final learning rate	Training time (hours)
180	0.0026	0.0029	0.00064	18.12

TABLE X
COMPARISON BETWEEN NSB AND LSTM-RNN

	NSB [Mean/Std]	LSTM-RNN [Mean/Std]
Divergence of main lobe (°)	0.437/0.323	0.446/0.324
Divergence of nulls (°)	0.000/0.000	0.072/0.111
SINR (dB)	27.196/2.970	27.159/2.973

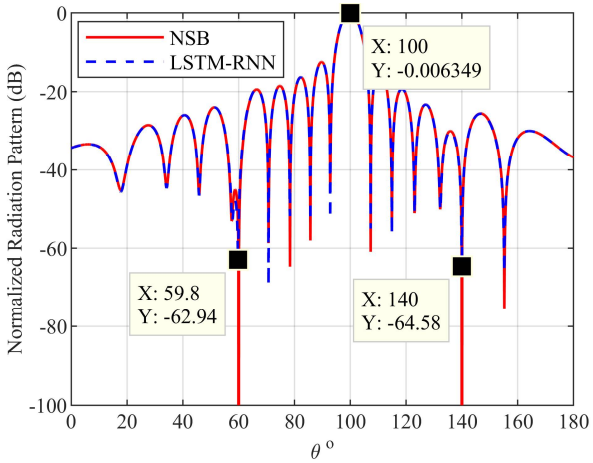
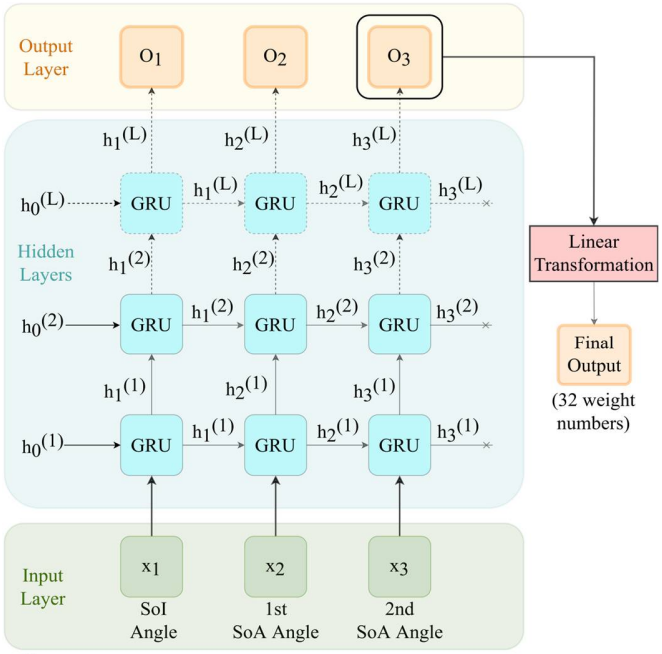


Fig. 8. Radiation patterns produced by NSB and LSTM-RNN for a SoI received at 100° and 2 SoAs received at respective AoAs equal to 60° and 140°.

Finally, an example of radiation patterns produced by the NSB algorithm and the LSTM-RNN for a SoI and two SoAs is given in Fig. 8. The radiation patterns seem to be almost identical, and they simply verify the above statistical analysis given in Table X.

B. *Use of GRUs*

GRUs are known to be a "lighter" version of LSTM units, because they use fewer gates and only a hidden state vector, as shown in Fig. 9. Their simpler architecture enables easier training and faster response than the LSTM units. Some studies have shown that this faster response does not affect their performance, while in some cases they are able to outperform the LSTM units [41].



$h_i^{(j)}$: Hidden State of time step i on j-th Layer

Fig. 9. L-layer GRU-RNN.

Due to their similar architecture, the same types of grid search are applied to find the most efficient GRU-RNN model. As

shown in Table XI, the most promising architectures are those with 4 hidden layers and hidden layer sizes equal to 512 and 1024. We choose a size equal to 512 for the same reason as in the case LSTM units.

TABLE XI
GRU-RNN GRID SEARCH (5-FOLD CROSS VALIDATION)

Hidden layers	Hidden Layer Size	64	128	256	512	1024
2	Training RMSE	0.0501	0.0508	0.0495	0.0487	0.0476
	Test RMSE	0.0502	0.0501	0.0514	0.0488	0.0482
3	Training RMSE	0.0497	0.0495	0.0473	0.0439	0.0397
	Test RMSE	0.0498	0.0503	0.0490	0.0458	0.0441
4	Training RMSE	0.0495	0.0490	0.0403	**0.0353**	**0.0323**
	Test RMSE	0.0497	0.0509	0.0443	**0.0415**	**0.0392**

Then, we look for the best hyperparameter combination by comparing different batch sizes and different learning rates. The results are shown in Table XII. Once again, as in the case of LSTM-RNNs, we avoid choosing low learning rates and low batch sizes to stay away from the same overfitting and increased training time problems. Therefore, we have chosen a learning rate equal to 0.001 and a batch size equal to 256.

TABLE XII
BATCH SIZE COMPARISON OF GRU-RNNS FOR VARIOUS LEARNING RATES (3-FOLD CROSS VALIDATION)

Batch Size	Learning Rate	0.001	0.003	0.005
32	Training RMSE	0.0241	0.0304	0.0373
	Test RMSE	0.0405	0.0419	0.0455
128	Training RMSE	0.0310	0.0287	0.0318
	Test RMSE	0.0419	0.0409	0.0472
256	Training RMSE	**0.0377**	0.0364	0.0366
	Test RMSE	**0.0414**	0.0465	0.0471
512	Training RMSE	0.0475	0.0414	0.0390
	Test RMSE	0.0494	0.0435	0.0489

After we have found the best settings for the GRU-RNN, we proceed to train and test the final model using the big dataset (5×10^6 records). The major part of this set, i.e., 4.9×10^6 records, is used for training, and the remainder, i.e., 10^5 records, is used for testing. The results are shown in Fig. 10 and Table XIII. First of all, it can be observed that the training and test RMSEs achieved by the GRU-RNN are approximately the same as those achieved by the LSTM-RNN. The main difference is that the training of the GRU-RNN takes much less time than the training of the LSTM-RNN, proving that the GRUs are as efficient as the LSTM units but with lower complexity, thus resulting in lower training time.

Next, the GRU-RNN is tested in terms of accuracy in the produced radiation patterns. The test is applied on 10^5 triads of AoAs (every triad corresponds to a SoI and two SoAs), on which the GRU-RNN has no prior "experience". A statistical analysis of the results derived from this test is given in Table XIV. It seems that the GRU-RNN provides high accuracy regarding both the main lobe steering and the nulls placement. This accuracy is also verified by the mean value of the SINR, which is almost identical to the respective SINR value achieved by the NSB algorithm. In addition, the mean values of the main lobe divergence and the nulls divergence derived by the GRU-RNN satisfy the respective requirements (i.e., main lobe divergence < 0.5° and nulls divergence < 0.1° as defined in section V).

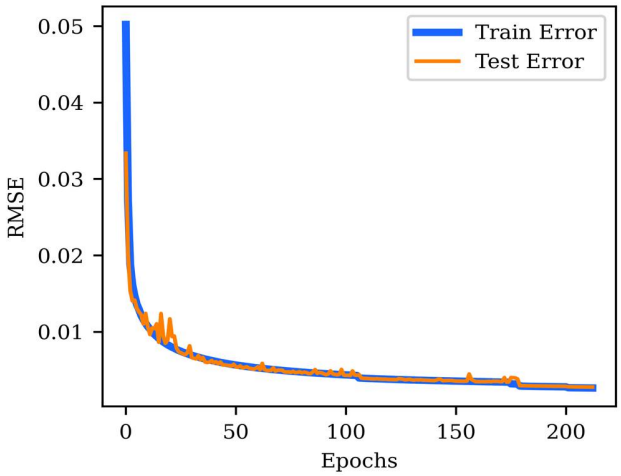


Fig. 10. Learning curves of the GRU-RNN.

TABLE XIII
GRU-RNN TRAINING AND TEST RESULTS

Epochs	Training RMSE	Test RMSE	Final learning rate	Training time (hours)
212	0.00262	0.00274	0.000512	14.83

TABLE XIV
COMPARISON BETWEEN NSB AND GRU-RNN

	NSB [Mean/Std]	GRU-RNN [Mean/Std]
Divergence of main lobe (°)	0.437/0.323	0.435/0.323
Divergence of nulls (°)	0.000/0.000	0.076/0.117
SINR (dB)	27.196/2.970	27.156/2.976

Finally, an example of radiation patterns produced by the NSB algorithm and the GRU-RNN for a SoI and two SoAs is given in Fig. 11. The radiation patterns seem to be identical, and they simply verify the above statistical analysis given in Table XIV.

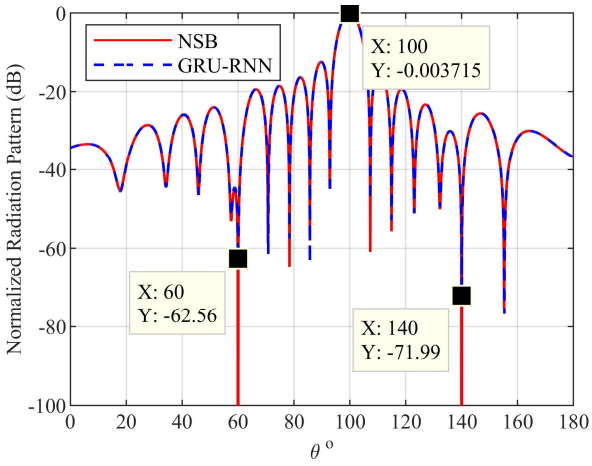


Fig. 11. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 2 SoAs received at respective AoAs equal to 60° and 140°.

VIII. Comparison Between NSB and NN Models

All the previous results are presented in summary in Table XV, to make it easier to compare and choose the best NN model. Algorithm complexity is a key indicator for demonstrating the computational advantage of the NN approach over the NSB model. By considering the number N of the incoming SoAs as the main complexity parameter, we calculated the complexity of each algorithm for a single input sample. The E_φ components needed by the NSB algorithm and the RNN parameters have already been imported and thus are not involved in the weight calculation process. Thus, the complexity of the NSB algorithm turns out to be $O(N^3) + O(N^2) + O(N) + O(1)$. On the other hand, the complexity of the FFNN and both RNN models is derived equal to $O(N) + O(1)$.

The mathematical simplicity of all NNs compared to the computationally expensive calculations performed by the NSB algorithm is reflected in the lower mean response time of all NN-based beamformers compared to that of the NSB algorithm. The mean response time is the mean value of the time required by each beamformer to extract the proper feeding weights. The same triads of incoming AoAs are used for all the beamformers. The measurements were made in the Google Colaboratory environment, using an Intel® Xeon® CPU @2.30GHz with 12GB of RAM (assigned by the Google Colaboratory environment), and they prove that all NN models are much faster than the NSB algorithm. The FFNN has the lowest response time but the highest divergence in terms of nulls placement. Therefore, the final choice has to be made between the LSTM-RNN and the GRU-RNN. As beamformers, both the LSTM-RNN and GRU-RNN models perform similarly with very high accuracy and much lower response time than that of the NSB algorithm. However, due to its significantly shorter training time, the GRU-RNN model is selected as the best possible solution.

TABLE XV
Comparison Between All NN Models and NSB

	Mean divergence of main lobe (°)	Mean divergence of nulls (°)	Mean SINR (dB)	Training time (hours)	Mean response time (sec)
NSB	0.437	0.000	27.196	-	1.47
FFNN	0.438	0.364	26.530	13.28	0.00033
LSTM-RNN	0.446	0.072	27.159	18.12	0.00481
GRU-RNN	0.435	0.076	27.156	14.83	0.00469

The "progressive adaptation" of the excitation weights at the output of the RNN model (discussed in the second paragraph of Section VII) can be verified by observing the radiation pattern produced by the RNN model at each time step. For the sake of simplicity, we consider again the case of Fig. 11 with a SoI received at 100° and 2 SoAs received at respective AoAs equal to 60° and 140°. This time, we store the model output for every time step (i.e., for every one of AoAs 100°, 60°, or 140° entering the RNN input), and we pass every output through the same linear transformation layer (shown in Fig. 9) to derive the excitation weights for every time step. Using these excitation weights, we produce respective radiation patterns (shown in Figs. 12 – 14) for every time step. The correct output is only derived at the end of the 3rd time step, i.e., when all three AoAs are taken into account (see Fig. 14).

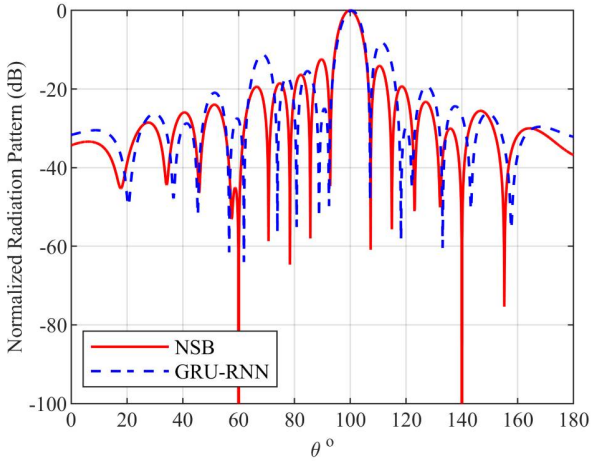


Fig. 12 Radiation pattern generated from the output of the GRU-RNN at the 1st time step.

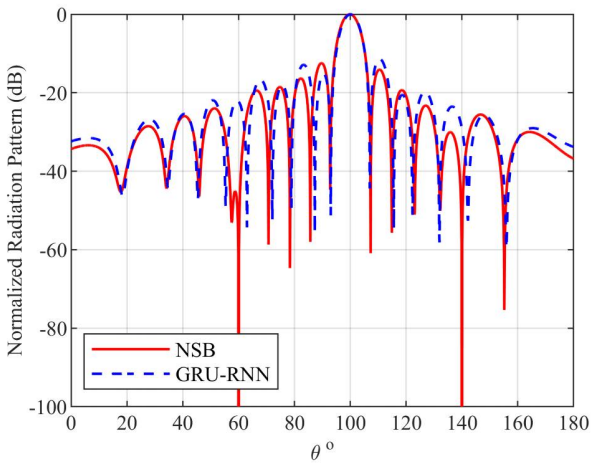


Fig. 13 Radiation pattern generated from the output of the GRU-RNN at the 2nd time step.

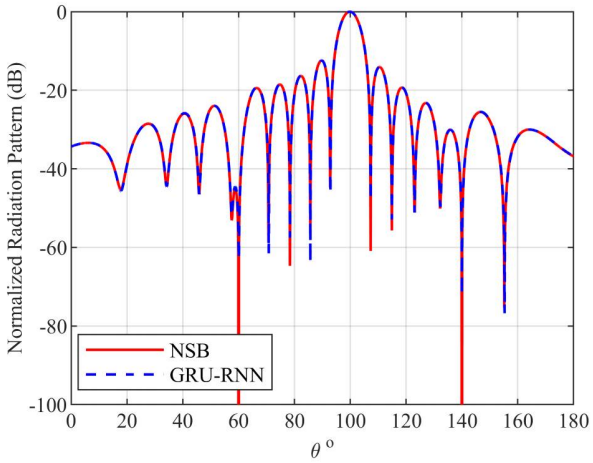


Fig. 14 Radiation pattern generated from the output of the GRU-RNN at the 3rd time step.

As we move from one time step to the next, the radiation pattern improves and looks more like that of the NSB algorithm. However, there is no correlation between a certain input value and an improvement in the radiation pattern. If such a correlation existed, the radiation pattern generated by the output at the second time step (Fig. 13) should have placed a null at 60° (AoA of the first SoA), thus verifying the intuition that the

output is "adapting" its value by placing respective nulls at AoAs of the new incoming SoAs. Nevertheless, this does not happen.

The important thing to notice here is that all these outputs are "interpreted" and transformed into weight vectors of size 32 as they pass through the linear transformation layer. This layer has been trained to transform the 512 numbers of the output at the last time step to a vector of size 32. There is no way to know which parts of the 512 output numbers correspond to the placement of the main lobe or a certain null, and it is unknown whether these parts are located in the same position for all the outputs of previous time steps. For these reasons, the radiation patterns of Figs. 12 – 14 only portray the fact that the RNN output improves as the model receives new information at each time step, thus verifying the initial motivation for the use of RNNs, as described in Section VII.

IX. Evaluation of the GRU-RNN Beamformer

In this section, we evaluate the performance of the GRU-RNN beamformer in comparison with the NSB algorithm for various numbers of SoAs. For this purpose, the NSB algorithm has been employed to produce datasets for cases of 2 to 10 SoAs, and then we use these datasets to train the GRU-RNN. Next, the GRU-RNN is tested in terms of accuracy in the produced radiation patterns. The test is applied on 10^5 combinations of AoAs (every combination corresponds to a SoI and N SoAs, where $N = 2, 3, \ldots, 10$), on which the GRU-RNN has no prior "experience". These combinations of AoAs are also applied to the NSB algorithm. The statistical analyses of the results derived by the NSB algorithm and the GRU-RNN beamformer are respectively given in Tables XVI and XVII.

TABLE XVI
NSB Performance for Various Numbers of SoAs

Number of SoAs	Divergence of main lobe (°) [Mean/Std]	Divergence of nulls (°) [Mean/Std]	SINR (dB) [Mean/Std]
2	0.437/0.323	0.000/0.000	27.196/2.970
3	0.427/0.317	0.000/0.000	27.276/2.931
4	0.420/0.310	0.000/0.000	27.331/2.892
5	0.416/0.305	0.000/0.000	27.369/2.882
6	0.413/0.301	0.000/0.000	27.405/2.872
7	0.412/0.296	0.000/0.000	27.425/2.869
8	0.412/0.292	0.000/0.000	27.427/2.899
9	0.418/0.290	0.000/0.000	27.397/2.947
10	0.425/0.288	0.000/0.000	27.338/3.046

TABLE XVII
GRU-RNN Performance for Various Numbers of SoAs

Number of SoAs	Divergence of main lobe (°) [Mean/Std]	Divergence of nulls (°) [Mean/Std]	SINR (dB) [Mean/Std]
2	0.435/0.323	0.076/0.117	27.156/2.976
3	0.427/0.317	0.098/0.155	27.192/2.951
4	0.422/0.313	0.095/0.157	27.229/2.916
5	0.415/0.306	0.096/0.160	27.245/2.917
6	0.415/0.303	0.096/0.163	27.260/2.918
7	0.412/0.297	0.110/0.187	27.216/2.931
8	0.418/0.296	0.127/0.223	27.107/3.003
9	0.418/0.290	0.141/0.234	26.990/3.127
10	0.428/0.291	0.152/0.261	26.823/3.279

A simple comparison between Tables XVI and XVII proves that the GRU-RNN beamformer is capable of providing high accuracy, even when increasing the number of SoAs. This is also verified by the radiation patterns of Figs. 15-22. The patterns have been produced by the NSB algorithm and the GRU-RNN beamformer for a SoI and various numbers of SoAs.

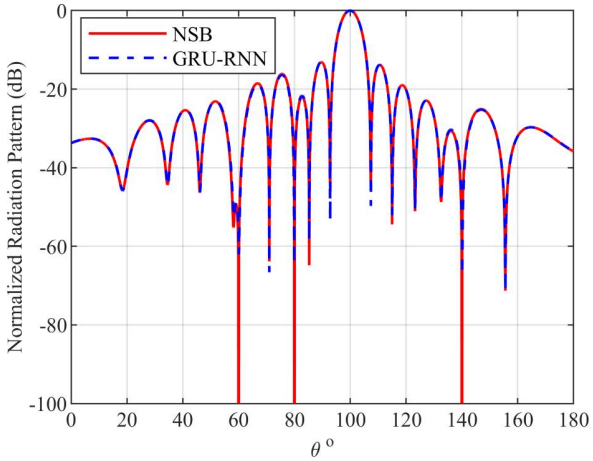


Fig. 15. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 3 SoAs received at respective AoAs equal to 60°, 80°, and 140°.

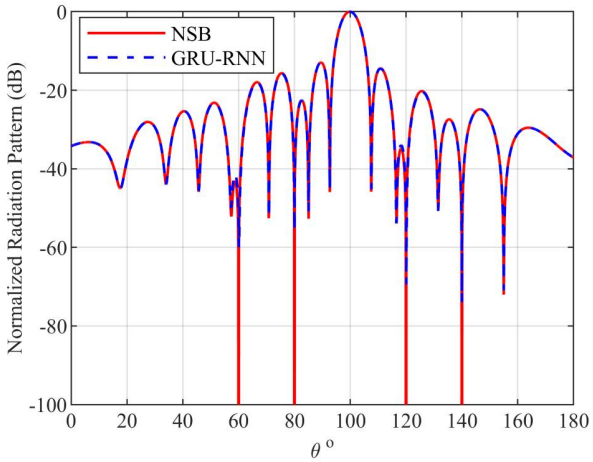


Fig. 16. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 4 SoAs received at respective AoAs equal to 60°, 80°, 120°, and 140°.

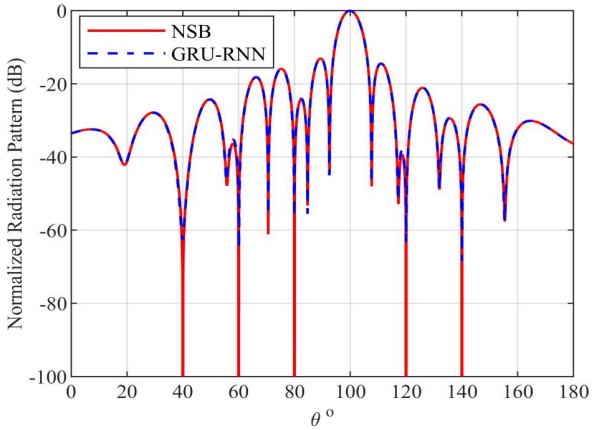


Fig. 17. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 5 SoAs received at respective AoAs equal to 40°, 60°, 80°, 120°, and 140°.

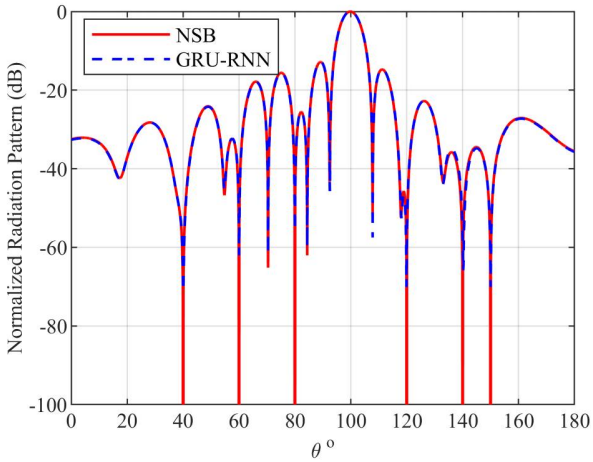


Fig. 18. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 6 SoAs received at respective AoAs equal to 40°, 60°, 80°, 120°, 140°, and 150°.

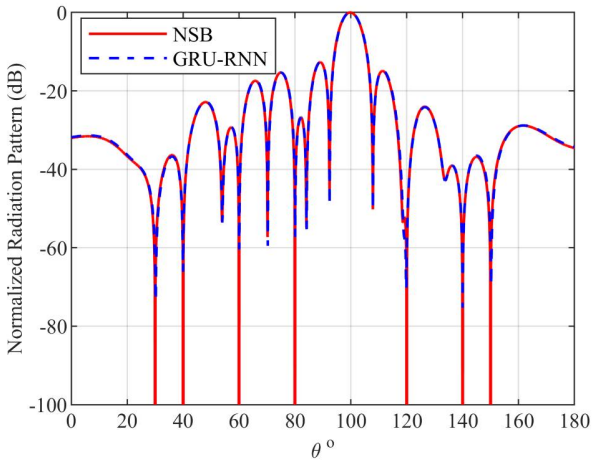


Fig. 19. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 7 SoAs received at respective AoAs equal to 30°, 40°, 60°, 80°, 120°, 140°, and 150°.

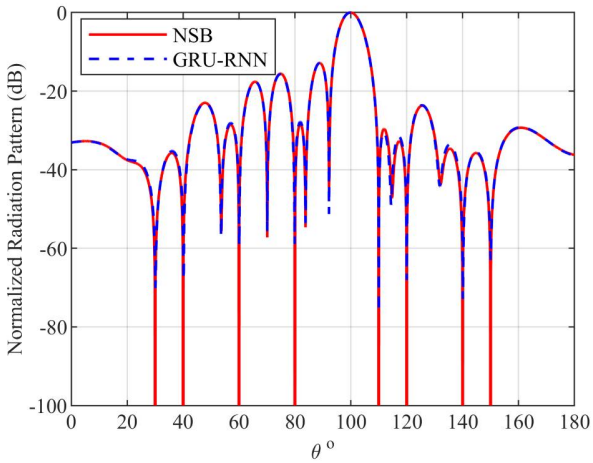


Fig. 20. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 8 SoAs received at respective AoAs equal to 30°, 40°, 60°, 80°, 110°, 120°, 140°, and 150°.

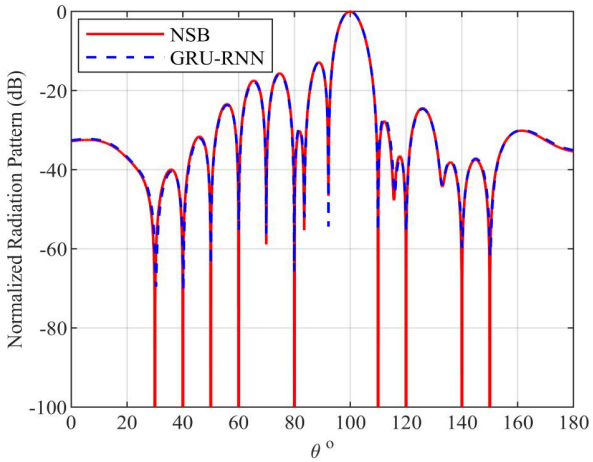


Fig. 21. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 9 SoAs received at respective AoAs equal to 30°, 40°, 50°, 60°, 80°, 110°, 120°, 140°, and 150°.

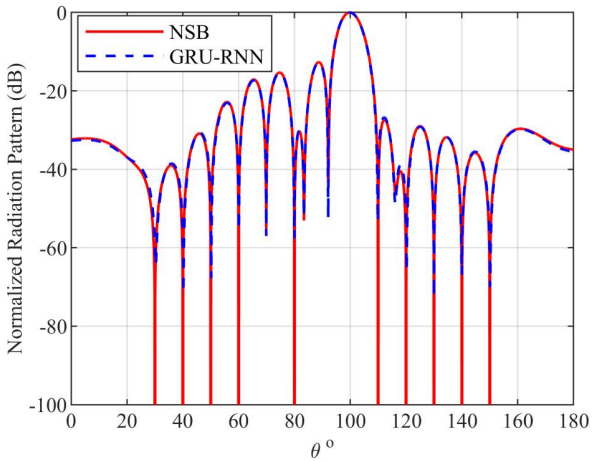


Fig. 22. Radiation patterns produced by NSB and GRU-RNN for a SoI received at 100° and 10 SoAs received at respective AoAs equal to 30°, 40°, 50°, 60°, 80°, 110°, 120°, 130°, 140°, and 150°.

X. Conclusions

The comparative results have shown that the GRU-RNN architecture, with four hidden layers, is the most optimal solution to the ABF problem, and has a similar performance compared to the LSTM-RNN architecture, but with less training time. The GRU-RNN model is capable of producing radiation patterns with an accuracy similar to that of radiation patterns produced by the NSB technique, thus meeting all the requirements defined in an environment with high noise conditions. To further demonstrate the capabilities of the GRU-RNN model as a beamformer, scenarios with different numbers of interference signals, from two to ten SoAs, have been implemented. After its training in all these different scenarios, the GRU-RNN model produces results with an accuracy similar to that of the NSB technique.

The improvement of the training process with different configurations, the exploitation of modified or even new optimization techniques and generally the investigation for a better and more efficient way of training could be the next steps in the current implementation of the GRU-RNN model, in order

to reduce the time cost of training. In this paper, we aimed exclusively at finding the NN model that meets the accuracy criteria of the produced radiation patterns. The reduction of the overall size of the model came second as a priority, although when we had to decide on the size of the hidden layers (i.e., 512 or 1024), we deliberately chose the computationally cheaper solution in terms of training time. However, smaller in size NN models could potentially achieve similar performance with less training time and less response time. Therefore, since the ABF is a real-time process performed in a real environment, the investigation of the most efficient NN model should aim not only at optimal performance but also at shorter training time and shorter response time.

This research focuses on finding the most appropriate NN type as an alternative ABF method. For this reason, it was considered good practice to approach the ABF problem in its simplest form. Of course, the simplest form of this problem is the application of the beamformer to a linear (1D) antenna array, as was done in this paper. Now that we are more confident about the NN type for the problem in question, we can extend this research to realistic 2D antenna arrays in the future. Since the proposed GRU-RNN model is capable of accurately mapping the incoming AoAs to appropriate excitation weights for 1D arrays, the usability of this model in 2D arrays is very high.

Finally, the ABF is just one of the two processes performed by a smart antenna to control the reception of incoming signals. The integration of a DoA estimation process into the current GRU-RNN model will significantly enhance the operation of smart antennas in practice.

REFERENCES

[1] I. P. Gravas, Z. D. Zaharis, T. V. Yioultsis, P. I. Lazaridis, and T. D. Xenos, "Adaptive beamforming with sidelobe suppression by placing extra radiation pattern nulls," *IEEE Trans. Antennas Propag.*, vol. 67, no. 6, pp. 3853–3862, June 2019.

[2] N. A. Sutton and D. S. Filipovic, "V-band monolithically integrated four-arm spiral antenna and beamforming network," presented at the *2012 IEEE Int. Symp. Antennas Propag.*, Chicago, IL, USA, July 2012.

[3] A. Young, M. V. Ivashina, R. Maaskant, O. A. Iupikov, and D. B. Davidson, "Improving the calibration efficiency of an array fed reflector antenna through constrained beamforming," *IEEE Trans. Antennas Propag.*, vol. 61, no. 7, pp. 3538–3545, July 2013.

[4] D. S. Prinsloo, M. V. Ivashina, R. Maaskant, and P. Meyer, "Beamforming strategies for active multi-mode antennas: Maximum gain, Signal-to-Noise ratio, and polarization discrimination," presented at the *Int. Conf. Electromagnetics in Advanced Applications (ICEAA)*, Palm Beach, Aruba, Aug. 2014.

[5] O. A. Iupikov, M. V. Ivashina, C. Cappellin, and N. Skou, "Digital-beamforming array antenna technologies for future ocean-observing satellite missions," presented at the *IEEE Int. Symp. Antennas Propag. (APSURSI)*, Fajardo, PR, USA, July 2016.

[6] O. Manoochehri, D. Erricolo, A. Darvazehban, and F. Monticone, "Design of compact beam-steering active slot antennas with a metasurface reflector," presented at the *United States National Committee of URSI National Radio Science Meeting (USNC-URSI NRSM)*, Boulder, CO, USA, Jan. 2019.

[7] P. Rocca, N. Anselmi, M. Salucci, G. Gottardi, L. Poli, and A. Massa, "A novel analytic beam steering approach for clustered phased array architectures," presented at the *IEEE Int. Symp. Antennas Propag. & USNC/URSI National Radio Science Meeting*, San Diego, CA, USA, July 2017.

[8] S. Zhou, F. Yang, S. Xu, and M. Li, "Beam nulling designs of reflectarray antenna using alternating projection method," presented at the *Cross Strait Radio Science & Wireless Technology Conf. (CSRSWTC)*, Fuzhou, China, Dec. 2020.

[9] C. Constantinides, S.K. Podilchak, S. Rotenberg, C. Mateo-Segura, G. Goussetis, J.-L. Gomez-Tornero, and G. Toso, "Leaky-Wave antenna with beam steering capability based on a meandered metallic waveguide," presented at the *15th European Conf. Antennas Propag. (EuCAP 2021)*, Dusseldorf, Germany, Mar. 2021.

[10] F. Vidal, H. Legay, G. Goussetis, and J.-P. Fraysse, "Joint precoding and resource allocation strategies applied to a large direct radiating array for GEO telecom satellite applications," presented at the *15th European Conf. Antennas Propag. (EuCAP 2021)*, Dusseldorf, Germany, Mar. 2021.

[11] O. Isik and K. P. Esselle, "Backward wave microstrip lines with complementary spiral resonators," *IEEE Trans. Antennas Propag.*, vol. 56, no. 10, pp. 3173–3178, Oct. 2008.

[12] T. Kokkinos and A. P. Feresidis, "Electrically small superdirective endfire arrays of metamaterial-inspired low-profile monopoles," *IEEE Antennas Wireless Propag. Lett.*, vol. 11, pp. 568–571, May 2012.

[13] T. Negishi, D. Erricolo, and P. L. E. Uslenghi, "Metamaterial spheroidal cavity to enhance dipole radiation," *IEEE Trans. Antennas Propag.*, vol. 63, no. 6, pp. 2802–2807, June 2015.

[14] A. H. El. Zooghby, C. G. Christodoulou, and M. Georgiopoulos, "Neural network-based adaptive beamforming for one- and two-dimensional antenna arrays," *IEEE Trans. Antennas Propag.*, vol. 46, no. 12, pp. 1891–1893, Dec. 1998.

[15] M. Abualhayja'a and M. Hussein, "Comparative study of adaptive beamforming algorithms for smart antenna applications," presented at the *Int. Conf. Communications, Signal Processing, and their Applications (ICCSPA)*, Sharjah, United Arab Emirates, Mar. 2021.

[16] Z. Zaharis, K. Gotsis, and J. N. Sahalos, "Comparative study of neural network training applied to adaptive beamforming of antenna arrays," *Prog. Electromagn. Res.*, vol. 126, pp. 269–283, Mar. 2012.

[17] X. Song, J. Wang, and X. Niu, "Robust adaptive beamforming algorithm based on neural network," presented at the *IEEE Int. Conf. Automation and Logistics*, Qingdao, China, Sep. 2008.

[18] Z. D. Zaharis, I. P. Gravas, P. I. Lazaridis, T. V. Yioultsis, C. S. Antonopoulos, and T. D. Xenos, "An effective modification of conventional beamforming methods suitable for realistic linear antenna arrays," *IEEE Trans. Antennas Propag.*, vol. 68, no. 7, pp. 5269–5279, July 2020.

[19] P. Kasemir, N. Sutton, M. Radway, B. Jeong, T. Brown, and D. S. Filipovič, "Wideband analog and digital beamforming," presented at the *9th Int. Conf. Telecommunication in Modern Satellite, Cable, and Broadcasting Services*, Nis, Serbia, Oct. 2009.

[20] G. Gottardi, L. Poli, P. Rocca, A. Montanari, A. Aprile, and A. Massa, "Optimal monopulse beamforming for side-looking airborne radars," *IEEE Antennas Wireless Propag. Lett.*, vol. 16, pp. 1221–1224, Nov. 2017.

[21] G. Gottardi, N. Ebrahimi, P. Rocca, and A. Massa, "Optimal synthesis of monopulse beamforming weights for airborne radars through convex optimization," presented at the *Int. Applied Computational Electromagnetics Society Symp. - Italy (ACES)*, Firenze, Italy, Mar. 2017.

[22] L. Poli, P. Rocca, G. Oliveri, and A. Massa, "Harmonic beamforming in time-modulated linear arrays," *IEEE Trans. Antennas Propag.*, vol. 59, no. 7, pp. 2538–2545, July 2011.

[23] N. Anselmi, P. Rocca, A. Massa, and E. Giaccari, "Synthesis of robust beamforming weights in linear antenna arrays," presented at the *IEEE Conf. Antenna Measurements & Applications (CAMA)*, Antibes Juan-les-Pins, France, Nov. 2014.

[24] N. Anselmi, P. Rocca, M. Salucci, and A. Massa, "Optimisation of excitation tolerances for robust beamforming in linear arrays," *IET Microw. Antennas Propag.*, vol.10, pp. 208–214, Jan. 2016.

[25] T. Shan, X. Pan, M. Li, S. Xu, and F. Yang, "Coding programmable metasurfaces based on deep learning techniques," *IEEE Trans. Emerg. Sel. Topics in Circuits and Systems*, vol. 10, no. 1, pp. 114–125, Mar. 2020.

[26] A. H. El Zooghby, C.G. Christodoulou, and M. Georgiopoulos, "A neural network-based smart antenna for multiple source tracking", *IEEE Trans. Antenna Propagat.*, vol. 48, no. 5, pp. 768–776, May 2000.

[27] M. Sarevska and A.-B. M. Salem, "Antenna array beamforming using neural network", *World Academy of Science, Engineering and Technology,* vol. 24, Jan. 2006.

[28] 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.

[29] Z. D. Zaharis *et al.*, "Implementation of antenna array beamforming by using a novel neural network structure," presented at the *Int. Conf.*

Telecommunications and Multimedia (TEMU), Heraklion, Greece, July 2016.
[30] H. Che, C. Li, X. He, and T. Huang, "A recurrent neural network for adaptive beamforming and array correction," *Neural Networks*, vol. 80, pp. 110–117, Apr. 2016.
[31] A. H. Sallomi and S. Ahmed, "Multi-layer feed forward neural network application in adaptive beamforming of smart antenna system," presented at the *Al-Sadeq Int. Conf. Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, Baghdad, Iraq, May 2016.
[32] X. Xiao and Y. Lu, "Data-based model for wide nulling problem in adaptive digital beamforming antenna array," *IEEE Antennas and Wireless Propag. Lett.*, vol. 18, no. 11, pp. 2249–2253, Nov. 2019.
[33] P. Ramezanpour, M. J. Rezaei, and M. R. Mosavi, "Deep-learning-based beamforming for rejecting interferences," *IET Signal Process.*, vol. 14, pp. 467–473, Sep. 2020.
[34] Y. Yu, H. Yin, J. Zhai, and C. Yu, "Behavioral modeling of millimeter wave beamforming transmitters with vector decomposition time delay recurrent neural network," presented at the *Int. Conf. Microwave and Millimeter Wave Technology (ICMMT)*, Shanghai, China, Sep. 2020.
[35] P. Bhadauria, R. Kumar, and S. Sharma, "Performance dependency of LSTM and NAR beamformers with respect to sensor array properties in V2I scenario", arXiv, 2021.
[36] D. Erricolo *et al*., "Machine learning in electromagnetics: A review and some perspectives for future research," presented at the *Int. Conf. Electromagnetics in Advanced Applications (ICEAA)*, Granada, Spain, Sep. 2019.
[37] K. S. Senthilkumar, K. Pirapaharan, P. R. P. Hoole, and H. R. H. Hoole, "Single perceptron model for smart beam forming in array antennas," *Int. Journal of Electrical and Computer Engineering*, vol. 6, no. 5, pp. 2300–2309, Oct. 2016.
[38] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, June 1997.
[39] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, pp. 1553–1565, Aug. 2014.
[40] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Advances in neural information processing systems*, pp. 2933–2941, June 2014.
[41] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: Taking yelp review dataset as an example," presented at the *Int. Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, Shanghai, China, June 2020.

Ioannis Mallioras (S'22) received the Integrated Master's Degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2021, where he is currently pursuing his Ph.D. degree. As of 2021, he is an early-stage researcher within the context of an Horizon 2020 Marie Skłodowska-Curie Innovative Training Networks Program entitled "Mobility and Training for beyond 5G Ecosystems (MOTOR5G)". The subject of his doctoral dissertation is machine learning algorithms for network prediction. His research interests include machine learning techniques, deep neural networks, beamforming and massive MIMO techniques, network traffic prediction, and optimization of network operations.

Zaharias D. Zaharis (M'13–SM'15) received the B.Sc. degree in physics, the M.Sc. degree in electronics, the Ph.D. degree in antennas and propagation modeling for mobile communications, and the Diploma degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1987, 1994, 2000, and 2011, respectively. From 2002 to 2013, he was with the Administration of the Telecommunications Network of the Aristotle University of Thessaloniki. Since 2013, he has been with the School of Electrical and Computer Engineering of the Aristotle University of Thessaloniki. He has been involved in several international research projects, such as EU Horizon 2020 MOTOR5G and RECOMBINE. He is the author of 73 scientific journal papers, 55 international conference papers, 5 book chapters, and one book. Recently, he was elected as Chairman of the Electron Devices / Microwave Theory and Techniques / Antennas and Propagation Joint Chapter of the IEEE Greece Section. His current research interests include design and optimization of antennas and microwave circuits, signal processing on smart antennas, development of evolutionary optimization algorithms, and neural networks. Dr. Zaharis is a member of the Technical Chamber of Greece, and is currently serving as an Associate Editor for IEEE ACCESS.

Pavlos I. Lazaridis (M'13–SM'15) received the Diploma degree in electrical engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1990, the M.Sc. degree in electronics from Université Pierre and Marie Curie (Paris 6), Paris, France, in 1992, and the Ph.D. degree from the École Nationale Supérieure des Télécommunications (ENST) Paris and Université Paris 6, in 1996. From 1991 to 1996, he was involved in research at France Télécom, and teaching at ENST Paris. In 1997, he became the Head of the Antennas and Propagation Laboratory, Télédiffusion de France/the France Télécom Research Center (TDF–C2R Metz). From 1998 to 2002, he was a Senior Examiner with the European Patent Office (EPO), The Hague, The Netherlands. From 2002 to 2014, he was involved in teaching and research with the ATEI of Thessaloniki, Thessaloniki, Greece, and Brunel University, London, U.K. He is currently a Professor of electronics and telecommunications with the University of Huddersfield, UK. He has been involved in several international research projects, such as EU Horizon 2020 MOTOR5G and RECOMBINE, NATO-SfP ORCA, and he has published over 150 research articles and several national and European patents. He is a member of the IET (MIET), Senior Member of URSI, and a Fellow of the Higher Education Academy (FHEA). He is currently serving as an Associate Editor for IEEE ACCESS.

Stelios Pantelopoulos received the Diploma degree in electrical engineering from the University of Patras, Patras, Greece, in 1990, and the M.Sc. degree in robotics (D.E.A. de robotique) from Université Pierre et Marie Curie - Paris VI of Paris, in 1992. From 1994 to 1997, he was with Athens Technology Center, Athens, Greece, first as a software developer and later as a researcher-analyst, while participating in commercial projects, and research and development projects. From 1997 to 1999, he was with Intrasoft, Athens, Greece, as a researcher-analyst. From 1999 to 2020, he was with SingularLogic, Athens, Greece, as the director of European projects. Since the end of 2021, he has been the director of the Greece branch of the Italian company Maggioli S.p.A. The branch belongs to the International Development Unit and acts as the Innovation Center of Maggioli Group participating in R&D funded European research and development projects, while at the same time dealing with the group's commercial activities in Greece and southeastern Europe.