informatics

MDPI

Article

GRU-Based Beam Pattern Synthesis for Optimized Uniform Linear Antenna Arrays

Armando Arce [1,*], Fernando Arce [2], Enrique Stevens-Navarro [3], Ulises Pineda-Rico [3], Mohammad Reza Rahmati [2] and Abel García-Barrientos [3]

[1] SECIHTI-Facultad de Ciencias, Universidad Autónoma de San Luis Potosí (UASLP), San Luis Potosí 78295, Mexico
[2] Centro de Investigaciones en Óptica (CIO), A.C., León 37150, Mexico; farce@cio.mx (F.A.); rahmati@cio.mx (M.R.R.)
[3] Facultad de Ciencias, Universidad Autónoma de San Luis Potosí (UASLP), San Luis Potosí 78295, Mexico; enrique.stevens@uaslp.mx (E.S.-N.); ulises.pineda@uaslp.mx (U.P.-R.); abel.garcia@uaslp.mx (A.G.-B.)
* Correspondence: armando.arce@uaslp.mx; Tel.: +52-444-826-2300 (ext. 5684)

Abstract

This study presents a deep learning-based framework for beam pattern synthesis in optimized uniform linear antenna arrays, combining Differential Evolution–based pre-optimization with recurrent neural network (RNN) modeling. Radiation patterns are first generated to satisfy sidelobe suppression and directivity constraints and are then used to train recurrent models that learn the mapping between radiation patterns and complex excitation parameters. A formal mathematical formulation of the Simple RNN, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) architectures is provided, together with a per–time-step computational cost analysis based on dominant matrix–vector multiplications. A comparative evaluation under identical training conditions shows that gated architectures significantly outperform the standard RNN. Although the LSTM achieves the lowest prediction errors, the GRU attains comparable performance with reduced structural complexity. Beam pattern synthesis experiments for unseen steering directions demonstrate accurate reconstruction of main lobe alignment, sidelobe levels (approximately −12 to −13 dB), and directivity values close to 8 dB. The floating-point operations (FLOPs) analysis indicates that the GRU requires fewer dominant operations per time step than the LSTM, potentially reducing computational cost and energy consumption in resource-constrained beamforming applications.

Keywords: beam pattern synthesis; uniform linear arrays; gated recurrent unit (GRU); differential evolution; recurrent neural networks (RNN); beam steering; antenna radiation patterns; data-driven modeling; computational cost analysis; energy-efficient beamforming

Check for updates

Academic Editor: Yang Shi

Received: 25 February 2026
Revised: 2 April 2026
Accepted: 10 April 2026
Published: 14 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Future generations of wireless networks, including 6G and beyond, demand communication solutions capable of ensuring high spectral efficiency, reliability, and adaptability under increasingly complex propagation conditions [1]. In this context, advanced transmission techniques such as beamforming and multiple-input multiple-output (MIMO) systems have emerged as key enabling technologies for meeting these performance requirements. Their effectiveness, however, relies heavily on the use of antenna arrays, which allow the controlled generation of directive beams toward specific users or spatial regions. Antenna arrays, composed of multiple radiating elements, provide the capability to shape and steer

radiation patterns by properly adjusting their excitation parameters. This spatial control is particularly relevant in environments affected by interference, scattering, and attenuation phenomena, where channel degradation can significantly compromise communication quality [2,3]. Such impairments are commonly encountered in satellite communications, radar systems, point-to-point links, and emerging mobile networks operating at high frequencies. By integrating antenna arrays into these systems, it becomes possible to regulate the direction, width, and amplitude distribution of radiation beams, thereby mitigating adverse propagation effects and enhancing overall system performance. Among the various array geometries, uniform linear arrays (ULAs) remain widely adopted due to their structural simplicity and analytical tractability.

Despite the maturity of array signal processing theory, beam pattern synthesis remains a computationally demanding task, particularly when stringent sidelobe and directivity requirements are imposed. Traditional deterministic formulations, including approaches based on Fourier techniques [4,5], provide analytical insight and structured solutions; however, they often rely on repeated evaluations of the array factor, which can become computationally intensive in practical optimization scenarios. To address the nonlinear and multimodal nature of the synthesis problem, evolutionary algorithms have been widely adopted as global optimization strategies. These methods offer flexibility and robustness when exploring complex search spaces, especially in antenna array design. Nevertheless, their performance typically depends on careful parameter tuning, and convergence toward high-quality solutions may require a substantial number of fitness evaluations [6–9]. As a consequence, the synthesis process can become computationally expensive, particularly when multiple steering directions or large datasets are required, motivating the exploration of alternative frameworks that emphasize both predictive capability and architectural efficiency.

Recent advances in deep learning (DL) have provided alternative strategies for addressing complex inverse problems across diverse scientific and engineering domains [10–12]. In electromagnetics, learning-based methodologies have been successfully applied to antenna characterization, design, and synthesis tasks, reducing the reliance on exhaustive electromagnetic simulations [13–15]. In particular, neural networks have demonstrated the ability to approximate nonlinear mappings between radiation patterns and antenna parameters [16,17].

While convolutional neural networks (CNNs) have frequently been employed for electromagnetic modeling [18–20], radiation patterns inherently exhibit sequential structure along the angular domain. Recurrent neural networks (RNNs), and especially gated architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are specifically designed to capture long-range dependencies in ordered data sequences. Their gating mechanisms enable effective modeling of complex dependencies while maintaining stable training dynamics [21–25].

In this work, a learning-based beam pattern synthesis framework for optimized uniform linear arrays is proposed. A Differential Evolution (DE) algorithm is first employed to generate high-quality radiation patterns that satisfy sidelobe and directivity constraints. These optimized patterns serve as training data for recurrent neural networks tasked with inferring the corresponding complex excitation vectors. A comparative analysis among a simple RNN, an LSTM, and a GRU model is conducted to evaluate their predictive capability and computational efficiency through both numerical experiments and a formal per–time-step computational cost analysis based on dominant matrix–vector multiplications. Particular emphasis is placed on the balance between accuracy and architectural complexity, as reduced model complexity may translate into lower computational cost and energy consumption in practical beamforming applications.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation, the recurrent neural network modeling framework and the associated computational cost analysis. Section 3 describes the antenna pre-optimization stage, the experimental setup, and the numerical analysis of the recurrent models. Section 4 discusses the obtained results and their practical implications. Finally, Section 5 summarizes the main conclusions of this study.

2. Problem Definition and Learning Framework

2.1. Linear Array Optimization Model

We consider a uniform linear antenna array composed of N omnidirectional radiating elements placed along the x-axis and operating in the azimuthal plane. The radiation behavior of the array is described through its array factor, which depends on both the spatial configuration of the elements and their associated complex excitations. For a given excitation vector $\boldsymbol{I}$, the array factor can be expressed as [26]:

$$AF(\theta, \boldsymbol{I}) = \sum_{n=1}^{N} I_n \exp(jkd_n \cos\theta - jkd_n \cos\theta_0), \tag{1}$$

where $I_n = A_n e^{j\xi_n}$ denotes the complex excitation of the n-th element, with amplitude A_n and phase ξ_n, k is the wavenumber, d_n represents the position of the n-th radiator along the array, and θ_0 is the steering direction.

In this framework, the complex excitation of each array element is parameterized by its amplitude and phase components. Beam steering toward a desired direction is achieved by introducing a phase reference associated with the steering angle, while the wavenumber and inter-element spacing determine the spatial phase distribution across the array. This formulation provides a compact representation of the radiation pattern generated by arbitrary excitation vectors and serves as the basis for both the optimization procedure and the subsequent learning stage.

The beam pattern synthesis task addressed in this work aims at generating radiation patterns that achieve high directivity toward a predefined look direction while maintaining controlled sidelobe behavior. The optimized radiation patterns obtained at this stage are subsequently used to construct the dataset required for training the recurrent neural network. From this perspective, the antenna array optimization problem can be formulated as the search for excitation vectors within a feasible amplitude–phase domain that satisfy the desired performance criteria.

Accordingly, the synthesis problem can be cast as the following constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{I}} \quad & \left\{ \frac{|AF(\theta_{SL}, \boldsymbol{I})|}{|AF(\theta_{\max}, \boldsymbol{I})|} \right\} + \left\{ \frac{1}{D(\theta, \boldsymbol{I})} \right\} \\ \text{s.t.} \quad & |AF(\theta_{\max}, \boldsymbol{I})| = 1, \\ & \boldsymbol{I} = \left\{ A_n e^{j\xi_n} \mid \boldsymbol{I} \in \mathbb{I} \right\}, \quad \forall n \in N, \end{aligned} \tag{2}$$

where θ_{SL} denotes the angular location of the maximum sidelobe, $\theta_{\max}$ corresponds to the main beam direction, $D(\theta, \boldsymbol{I})$ represents the directivity associated with the excitation vector $\boldsymbol{I}$, and $\mathbb{I}$ defines the feasible amplitude–phase domain.

Due to the nonlinear and multimodal nature of the antenna synthesis problem, population-based evolutionary strategies are well suited to explore the solution space and generate representative excitation–pattern pairs. The numerical implementation of this optimization stage and its role as a precursor for data-driven beam pattern inference are detailed in the following section.

2.2. Recurrent Neural Network–Based Modeling

For the beam pattern synthesis task in linear antenna arrays, a Gated Recurrent Unit (GRU) network is employed. GRUs belong to the family of recurrent neural networks (RNNs) and are specifically designed to model sequential data, such as time series, audio signals, or ordered feature representations. In the context of this work, the GRU is used to capture the underlying sequential structure present in the radiation pattern representations generated during the pre-optimization stage.

A key advantage of GRUs lies in their ability to model long-term dependencies while maintaining a relatively simple internal structure. Unlike Long Short-Term Memory (LSTM) networks, which rely on three gating mechanisms, GRUs employ only two gates: an update gate and a reset gate. This reduced architectural complexity results in lower computational cost and faster training times, without a significant loss in performance for many sequential learning tasks [27,28].

Comparative studies have shown that GRU-based models achieve performance levels comparable to those of LSTM networks across a wide range of applications, including speech recognition, language modeling, and time-series prediction [29,30]. Furthermore, their compact structure makes GRUs particularly suitable for scenarios where computational efficiency and training stability are important considerations, such as data-driven electromagnetic modeling.

The gating mechanisms within a GRU dynamically regulate the flow of information through time. The update gate controls how much of the previous hidden state is retained in the current state, enabling the network to preserve relevant long-term information. Conversely, the reset gate determines the extent to which past information is disregarded when computing the candidate hidden state, allowing the model to focus on recent input patterns when appropriate [31]. Together, these mechanisms enable the GRU to effectively capture both short- and long-range dependencies in sequential data.

The base architecture used for the Simple RNN, GRU, and LSTM models in this study consists of two stacked recurrent layers with 1024 and 512 hidden units, respectively. Both layers employ the hyperbolic tangent (tanh) activation function, as shown in Figure 1. A batch normalization layer is subsequently applied to improve training stability, followed by two fully connected layers with ReLU activation functions. Dropout layers are interleaved to mitigate overfitting. The network concludes with a linear output layer, which produces the final beam pattern representation inferred from the input sequence. The proposed architecture is the result of a systematic hyperparameter fine-tuning process.

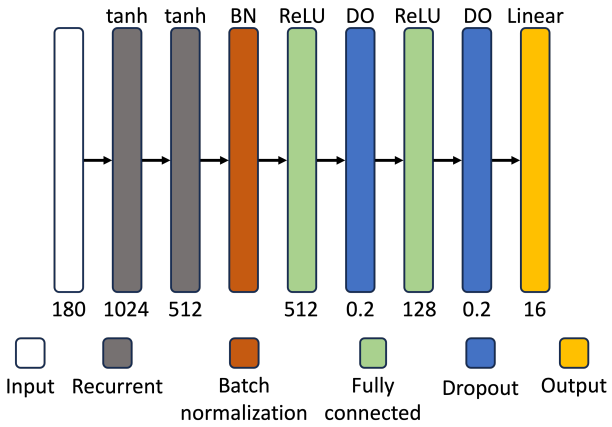


Figure 1. Recurrent neural network architecture adopted for beam pattern synthesis in uniform linear antenna arrays. The two recurrent layers (in gray) are instantiated as Simple recurrent neural network (RNN), Gated Recurrent Unit (GRU), or Long Short-Term Memory (LSTM) units depending on the model, while the remaining layers are shared across all configurations.

2.3. Mathematical Formulation and Computational Cost of Recurrent Architectures

In order to provide a formal comparison among the considered recurrent models, this section presents the mathematical formulation of the Simple RNN, GRU, and LSTM architectures, followed by a per–time-step computational cost analysis based on dominant matrix–vector multiplications.

Let an input sequence be defined as $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, where $x_t \in \mathbb{R}^I$ denotes the input vector at time step t, and T is the sequence length. Each recurrent architecture generates hidden states $h_t \in \mathbb{R}^H$, which propagate temporal information across time steps. The network output at each time step is denoted as $\widehat{y}_t \in \mathbb{R}^O$.

2.3.1. Simple Recurrent Neural Network

The Simple RNN employs an input-to-hidden weight matrix $W_x \in \mathbb{R}^{H\times I}$, a hidden-to-hidden weight matrix $W_h \in \mathbb{R}^{H\times H}$, and an output weight matrix $W_y \in \mathbb{R}^{O\times H}$. Additionally, the network includes two bias vectors: a hidden bias $b_h \in \mathbb{R}^H$ and an output bias $b_y \in \mathbb{R}^O$.

At each time step t, the hidden state is computed by forming the linear combination $a_t = W_x x_t + W_h h_{t-1} + b_h$, where W_x processes the current input vector x_t, W_h propagates information from the previous hidden state h_{t-1}, and b_h provides an affine offset. This intermediate representation is passed through a nonlinear activation function to obtain $h_t = \phi(a_t)$, enabling nonlinear temporal modeling.

The network output is computed through the linear transformation $o_t = W_y h_t + b_y$, followed by a task-dependent activation function $\widehat{y}_t = \psi(o_t)$, where $\psi(\cdot)$ depends on the learning objective (e.g., regression or classification).

Therefore, the complete set of feedforward equations for the Simple RNN at time step t is given by:

$$a_t = W_x x_t + W_h h_{t-1} + b_h, \tag{3}$$

$$h_t = \phi(a_t), \tag{4}$$

$$o_t = W_y h_t + b_y, \tag{5}$$

$$\widehat{y}_t = \psi(o_t). \tag{6}$$

This architecture shares the weight matrices W_x, W_h, and W_y, as well as the bias vectors b_h and b_y, across all time steps [32–36]. The recurrence mechanism is implemented through the hidden states $h_t \in \mathbb{R}^H$, which enable sequential memory propagation over time without the use of gating mechanisms. Although computationally efficient, this structure may struggle to model long-term dependencies due to gradient attenuation across extended sequences.

2.3.2. Gated Recurrent Unit (GRU) Network

The mathematical model of a standard GRU network is defined by a set of weight matrices and bias vectors governing its internal gating mechanisms. The update gate employs matrices $W_z \in \mathbb{R}^{H\times I}$ and $U_z \in \mathbb{R}^{H\times H}$, while the reset gate uses $W_r \in \mathbb{R}^{H\times I}$ and $U_r \in \mathbb{R}^{H\times H}$. The candidate hidden state is computed through $W_h \in \mathbb{R}^{H\times I}$ and $U_h \in \mathbb{R}^{H\times H}$. The network output is produced using $W_y \in \mathbb{R}^{O\times H}$. Corresponding bias vectors include b_z, b_r, $b_h \in \mathbb{R}^H$, and $b_y \in \mathbb{R}^O$.

At each time step t, the update gate is computed as $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$, where $\sigma(\cdot)$ denotes the sigmoid activation function and regulates how much past information is retained. The reset gate is defined as $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$, controlling the contribution of previous memory when forming the candidate hidden state.

Using the reset gate, the candidate hidden state is calculated as $\tilde{h}_t = \tanh\big(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h\big)$, where $\odot$ denotes element-wise multiplication. The hidden-state update

is then obtained through the convex combination $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$, enabling adaptive memory retention. Finally, the output is computed as $o_t = W_y h_t + b_y$, followed by the task-dependent activation $\widehat{y}_t = \psi(o_t)$.

Therefore, the complete set of feedforward equations for the GRU at time step t is given by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{7}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{8}$$
$$\tilde{h}_t = \tanh\big(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h\big), \tag{9}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{10}$$
$$o_t = W_y h_t + b_y, \tag{11}$$
$$\widehat{y}_t = \psi(o_t). \tag{12}$$

A GRU computes its hidden state $h_t \in \mathbb{R}^H$ by combining past memory with the current input through the reset and update gates [28,33–35,37]. The additive update structure promotes stable gradient propagation across long sequences while requiring fewer gating components than the LSTM architecture. This reduced structural complexity results in lower computational cost per time step compared to LSTM networks, while maintaining strong capability for modeling long-term dependencies.

2.3.3. Long Short-Term Memory (LSTM) Network

The LSTM architecture extends recurrent modeling by introducing three gating mechanisms together with an internal cell state. The network employs input-to-hidden weight matrices W_i, W_f, W_o, and $W_c \in \mathbb{R}^{H \times I}$ and hidden-to-hidden matrices U_i, U_f, U_o, and $U_c \in \mathbb{R}^{H \times H}$, associated with bias vectors b_i, b_f, b_o, and $b_c \in \mathbb{R}^H$. The output transformation is defined by $W_y \in \mathbb{R}^{O \times H}$ and $b_y \in \mathbb{R}^O$.

At each time step t, the input gate is computed as $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$, regulating how much new information enters the cell state. The forget gate is defined as $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$ and determines how much of the previous cell state C_{t-1} is retained. The output gate is obtained as $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$, controlling the exposure of internal memory to the hidden representation.

The candidate cell state is computed as $\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$. The memory update is then performed through $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$, enabling controlled long-term information propagation. The hidden state is finally obtained as $h_t = o_t \odot \tanh(C_t)$. As in previous architectures, the output is computed as $o_t^{(out)} = W_y h_t + b_y$, followed by the task-dependent activation $\widehat{y}_t = \psi(o_t^{(out)})$.

Therefore, the complete set of feedforward equations for the LSTM at time step t is given by:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{13}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{14}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{15}$$
$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{16}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{17}$$
$$h_t = o_t \odot \tanh(C_t), \tag{18}$$
$$o_t^{(out)} = W_y h_t + b_y, \tag{19}$$
$$\widehat{y}_t = \psi(o_t^{(out)}). \tag{20}$$

The LSTM maintains a separate cell state $C_t \in \mathbb{R}^H$ that enables robust long-term memory retention and mitigates gradient vanishing issues [33–36]. The use of three gating mechanisms provides fine-grained control over information flow; however, this increased architectural complexity results in higher computational cost per time step compared to both the Simple RNN and the GRU.

2.3.4. Computational Cost Analysis per Time Step

The computational cost of recurrent neural networks is primarily dominated by matrix–vector multiplications, which account for most of the floating-point operations (FLOPs). In contrast, bias additions, nonlinear activation functions, and element-wise operations contribute comparatively little and are typically neglected in complexity analyses. Furthermore, the output layer is excluded from this evaluation, since its cost depends on the specific learning task. Throughout this analysis, I, H, and O denote the input, hidden, and output dimensions, respectively.

Simple RNN

For the Simple RNN, the dominant operations at each time step arise from the input transformation $W_x x_t$, where $W_x \in \mathbb{R}^{H\times I}$, and the recurrent transformation $W_h h_{t-1}$, where $W_h \in \mathbb{R}^{H\times H}$.

The multiplication $W_x x_t$ requires approximately $2IH$ FLOPs, while $W_h h_{t-1}$ requires approximately $2H^2$ FLOPs. Therefore, the total computational cost per time step is

$$\mathcal{C}_{\text{RNN}} = 2IH + 2H^2 = 2(I + H)H.$$

GRU

In the GRU architecture, three gated components are computed at each time step: the update gate, the reset gate, and the candidate hidden state. Each component performs two matrix–vector multiplications: one input contribution and one recurrent contribution. Thus, each component requires approximately $2IH + 2H^2 = 2(I + H)H$ FLOPs. Since three such components are evaluated per time step, the total cost becomes

$$\mathcal{C}_{\text{GRU}} = 3 \cdot 2(I + H)H = 6(I + H)H.$$

LSTM

Similarly, the LSTM computes four gated components at each time step: the input gate, forget gate, output gate, and candidate cell state. Each component requires two matrix–vector multiplications, yielding a cost of $2(I + H)H$ FLOPs per component. Consequently, the total computational cost per time step is

$$\mathcal{C}_{\text{LSTM}} = 4 \cdot 2(I + H)H = 8(I + H)H.$$

Comparative Analysis

From a per–time-step computational perspective, the recurrent architectures exhibit increasing complexity due to their structural design,

$$\mathcal{C}_{\text{RNN}} < \mathcal{C}_{\text{GRU}} < \mathcal{C}_{\text{LSTM}}.$$

Although the GRU and LSTM provide improved capability for modeling long-term dependencies compared to the Simple RNN, this enhancement comes at the expense of higher computational cost. Notably, the GRU reduces the per–time-step FLOPs by approximately 25% compared to the LSTM (i.e., $6(I + H)H$ versus $8(I + H)H$), which partially

explains the improved computational efficiency observed in practice, as summarized in Table 1.

Table 1. Per–time-step computational cost (FLOPs) of recurrent architectures considering dominant matrix–vector multiplications.

Architecture	# Components	Cost per Component	Total FLOPs per Time Step
Simple RNN	1	$2(I+H)H$	$2(I+H)H$
GRU	3	$2(I+H)H$	$6(I+H)H$
LSTM	4	$2(I+H)H$	$8(I+H)H$

3. Experimental Setup and Numerical Analysis

3.1. Antenna Pre-Optimization Stage

Before training the recurrent models, we performed a preliminary beam pattern synthesis stage using Differential Evolution (DE). Widely adopted in electromagnetics (including beam pattern synthesis and antenna design), DE is a robust population-based stochastic optimizer characterized by a small set of control parameters, straightforward real-valued operators, and a balance between exploration and exploitation in continuous spaces. In our setting, DE evaluates and iteratively refines candidate complex excitations to obtain quasi-optimal radiation patterns that satisfy prescribed directivity and isolation level requirements.

It is important to emphasize that the DE-based pre-optimization stage is not an optional component of the proposed framework, but rather the mechanism by which high-quality, constraint-satisfying training data are generated. The beam pattern synthesis problem addressed here does not admit an analytical solution. The task of inferring complex excitation vectors that simultaneously satisfy sidelobe suppression and directivity constraints constitutes a nonlinear, multimodal inverse problem for which no exact data generation procedure exists. Accordingly, DE is employed to produce radiation patterns that encode the desired electromagnetic design criteria directly at the data generation level. This has two important consequences for the learning stage. First, the input space presented to the network is physically consistent, as all training patterns represent valid solutions to the synthesis problem, enabling effective generalization to unseen steering directions. Second, by restricting the training data to solutions satisfying the prescribed constraints, the DE stage reduces the inherent ambiguity of the one-to-many inverse mapping between radiation patterns and excitation vectors, facilitating stable convergence of the recurrent models.

For the numerical analysis, a uniform linear antenna array of eight elements and inter-element spacing $d = 0.5\lambda$ was used as the reference geometry (Figure 2). The array factor response on the azimuthal plane was assessed across a predefined steering window of $50^\circ \leq \theta \leq 130^\circ$ with 10° increments; the training set therefore contained 847 instances per pointing angle, yielding a total of 7623 radiation patterns.

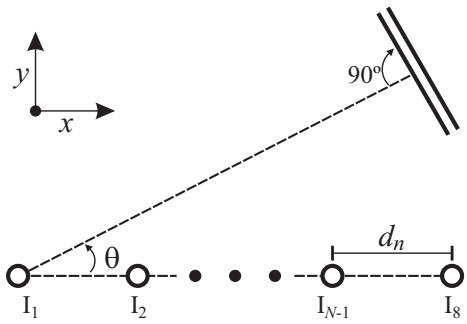


Figure 2. Geometry of the eight-element uniform linear antenna array used in the proposed framework.

The size of the training set was determined through an iterative process. Initial experiments with a reduced number of training patterns resulted in overfitting across all evaluated architectures, despite hyperparameter tuning and regularization strategies. Training patterns were progressively added until stable generalization was achieved. The current dataset therefore represents the minimum coverage required for the recurrent models to learn a reliable mapping between radiation patterns and excitation vectors within the considered steering window. Furthermore, the stochastic nature of DE ensures that each independent run produces a different solution, naturally generating a range of radiation patterns with varying sidelobe level and directivity trade-offs for each steering direction, resulting in a representative and diverse set of training samples.

Optimization relied on the *DE/rand/1/bin* strategy, using a mutation factor $F = 0.5$ and binomial crossover with $CR = 0.9$; selection followed a best-of-three tournament in which the lowest-cost candidate survived. The algorithm was initialized with a population of 500 individuals and evolved for 350 generations to ensure a comprehensive sampling of the search space. All pre-synthesis experiments were conducted in MATLAB R2019b.

The selected parameter configuration follows well-established convergence guidelines for DE [38,39]. A mutation factor $F = 0.5$ lies within the range recommended for balancing population diversity and selective pressure, while a crossover rate $CR = 0.9$ promotes sufficient recombination to sustain exploratory behavior across generations. The population size of 500 individuals is consistent with the rule-of-thumb of 5–10 individuals per search-space dimension for an 8-element array with amplitude and phase degrees of freedom. Under this configuration, 350 generations provide adequate budget for reliable convergence toward high-quality excitation vectors. Consequently, every radiation pattern in the resulting dataset satisfies the directivity and sidelobe constraints imposed by the fitness function in (2), ensuring that the training data reflects meaningful and physically valid beamforming solutions.

3.2. Training and Evaluation of Recurrent Neural Models

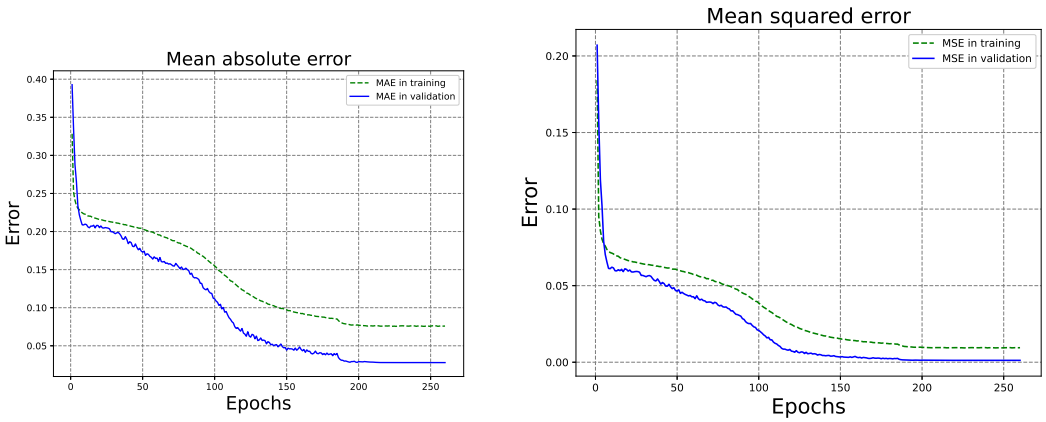
After the antenna pre-optimization stage, the generated radiation patterns were organized to construct the dataset used for training and evaluating the recurrent neural models. Each radiation pattern was represented as a one-dimensional sequence corresponding to the normalized array factor samples along the azimuthal plane. These sequences serve as the network inputs, while the associated amplitude and phase excitation vectors constitute the target outputs. The dataset was partitioned into training and validation subsets following the same protocol for all evaluated architectures to ensure a fair comparison.

Three recurrent neural models were considered in this study: a simple recurrent neural network (RNN), a Long Short-Term Memory (LSTM) network, and a Gated Recurrent Unit (GRU) network. All models were trained under identical conditions using the mean squared error (MSE) as the loss function and the Adam optimization algorithm. This setup allows the observed performance differences to be attributed primarily to the intrinsic modeling capabilities of each recurrent architecture rather than to variations in the training procedure.

Figure 3 illustrates the evolution of the training and validation errors for the GRU model in terms of mean absolute error (MAE) and mean squared error (MSE). As training progresses, both metrics exhibit a consistent decrease, indicating stable convergence and effective learning of the underlying radiation pattern representations. The close agreement between training and validation curves suggests that the GRU model generalizes well and does not suffer from overfitting.

Beyond accuracy metrics, architectural efficiency plays an important role in data-driven beam pattern inference. Compared to LSTM networks, GRU models employ a reduced gating structure, which results in a lower computational complexity per time

step. This reduction translates into fewer arithmetic operations during both training and inference, leading to shorter convergence times and, in practical implementations, reduced computational and energy demands. Although the LSTM achieves slightly lower error values, the GRU offers a more favorable balance between predictive performance and computational efficiency, which is particularly relevant for large-scale simulations and resource-constrained deployment scenarios.



(**a**) (**b**)

Figure 3. Training and validation performance of the GRU model. (**a**) Mean absolute error (MAE) during training and validation. (**b**) Mean squared error (MSE) during training and validation.

A quantitative comparison of the three recurrent architectures is presented in Table 2. The simple RNN exhibits the highest error levels among the evaluated models, reflecting its limited capability to capture long-range dependencies in sequential data. The LSTM achieves the lowest MSE and MAE values during both training and validation, highlighting its strong representational power. The GRU model attains performance comparable to that of the LSTM, with slightly higher error values but still outperforming the standard RNN.

Table 2. Comparative performance of the recurrent neural network models.

Architecture	MSE Training	MSE Validation	MAE Training	MAE Validation
RNN	4.2103×10^{-3}	1.7780×10^{-3}	0.0492	0.0323
LSTM	2.0915×10^{-3}	1.2811×10^{-3}	0.0348	0.0252
GRU	2.8561×10^{-3}	1.2199×10^{-3}	0.0409	0.0279

All numerical experiments were conducted on a workstation equipped with an Intel(R) Xeon(R) CPU running at 2.00 GHz and 64 GB of RAM (ASUS, León, Mexico), together with an NVIDIA RTX 3090 graphics processing unit (NVIDIA, León, Mexico).

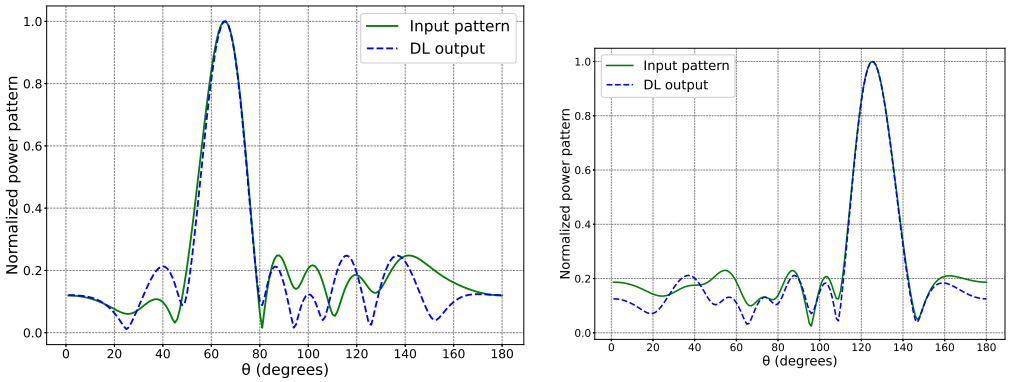
From a beam pattern synthesis perspective, the reported error metrics reflect the networks' ability to infer excitation parameters that reproduce radiation patterns satisfying the design constraints imposed during the pre-optimization stage. The reduced MSE and MAE values achieved by the LSTM and GRU models indicate that these architectures effectively capture the sequential structure embedded in the radiation patterns. In particular, the GRU provides a favorable trade-off between accuracy and architectural simplicity, making it an attractive alternative for data-driven beam pattern inference in linear antenna arrays.

3.3. Beam Pattern Synthesis with GRU Models

As part of the evaluation stage, the trained GRU model was assessed using radiation patterns corresponding to steering directions that were not included in the training dataset.

This evaluation strategy ensures that the reported results reflect the model's generalization capability rather than simple interpolation over previously seen patterns. In particular, the evaluation patterns were selected at angular locations that differ from those used during training, which was conducted within the range of $50^\circ \leq \theta \leq 130^\circ$ using 10° phase increments.

Figure 4 illustrates two representative evaluation cases, denoted as evaluation pattern 1 and evaluation pattern 2. In each figure, the original input radiation pattern is shown together with the beam pattern synthesized using the excitation parameters inferred by the GRU model. A close agreement between the target and synthesized patterns can be observed, especially in the main lobe region, where the beam direction and peak magnitude are accurately reproduced. Minor discrepancies appear mainly in the sidelobe regions, which is expected given the highly sensitive and nonlinear nature of sidelobe formation.



(**a**) (**b**)

Figure 4. Comparison between input radiation patterns and GRU-synthesized beam patterns for unseen steering directions. (**a**) Evaluation pattern 1 at $\theta = 65^\circ$, and (**b**) evaluation pattern 2 at $\theta = 125^\circ$.

The complex excitation parameters inferred by the GRU model for these evaluation cases are reported in Table 3. The table lists the amplitude and phase values associated with each antenna element of the uniform linear array. These inferred excitation vectors constitute the direct output of the GRU-based inference process and are subsequently used to reconstruct the corresponding radiation patterns through the array factor formulation.

Table 3. GRU-predicted amplitude and phase excitation values.

Antenna Element	Evaluation Pattern 1	Evaluation Pattern 2
1:	$6.61\angle 69.47^\circ$	$6.73\angle 69.72^\circ$
2:	$8.14\angle 66.36^\circ$	$5.56\angle 80.52^\circ$
3:	$9.94\angle 39.27^\circ$	$8.25\angle 84.79^\circ$
4:	$10.84\angle 67.73^\circ$	$11.99\angle 78.48^\circ$
5:	$6.79\angle 49.67^\circ$	$4.68\angle 79.53^\circ$
6:	$9.95\angle 33.09^\circ$	$9.47\angle 78.69^\circ$
7:	$7.64\angle 83.06^\circ$	$8.98\angle 52.82^\circ$
8:	$8.65\angle 62.13^\circ$	$6.91\angle 92.56^\circ$

To further quantify the quality of the synthesized beams, the resulting sidelobe level (SLL) and directivity values are summarized in Table 4. For both evaluation patterns, the GRU-generated beams achieve sidelobe levels on the order of -12 dB to -13 dB, while maintaining directivity values close to 8 dB. These results indicate that the inferred excitation parameters preserve the essential beamforming characteristics imposed during the pre-optimization stage.

Table 4. Sidelobe level and directivity inferred by the GRU model.

Evaluation Pattern 1		Evaluation Pattern 2	
SLL (dB)	***D* (dB)**	**SLL (dB)**	***D* (dB)**
−12.11	8.18	−13.49	7.76

To further assess the generalization capability of the proposed framework, Table 5 presents a statistical summary of the sidelobe level and directivity values obtained by the GRU model over the full evaluation range from 50° to 130° in 5° increments, covering both training and unseen steering directions. For each angle, the mean and standard deviation are computed from multiple independent synthesis runs. The results show that the GRU model achieves consistent performance across the entire range, with mean SLL values between approximately −10.97 dB and −13.83 dB and mean directivity values between 6.93 dB and 8.69 dB. Notably, the average performance over unseen angles (mean SLL = −12.42 dB, mean D = 8.23 dB) is statistically comparable to that over training angles (mean SLL = −12.46 dB, mean D = 8.03 dB), with a lower standard deviation in directivity (0.39 vs. 0.66 dB), indicating that the model generalizes consistently beyond the discrete steering directions used during training.

Table 5. Statistical summary of sidelobe level and directivity values predicted by the GRU model over the full steering range.

θ_0 (°)	Type [1]	Mean SLL (dB)	Std SLL (dB)	Mean D (dB)	Std D (dB)
50	T	−11.74	2.52	6.93	0.62
55	U	−12.60	1.05	7.73	0.10
60	T	−11.66	1.14	7.85	0.33
65	U	−12.25	0.32	8.21	0.08
70	T	−11.64	1.59	7.88	0.53
75	U	−13.83	0.72	8.44	0.20
80	T	−13.33	0.78	8.69	0.11
85	U	−11.26	1.11	8.56	0.23
90	T	−12.25	1.71	8.65	0.09
95	U	−10.97	0.40	8.60	0.04
100	T	−13.44	1.49	8.67	0.10
105	U	−13.62	0.50	8.56	0.10
110	T	−11.99	0.13	8.27	0.18
115	U	−11.35	1.95	8.05	0.32
120	T	−13.53	1.88	8.02	0.14
125	U	−13.44	0.15	7.67	0.15
130	T	−12.55	0.49	7.32	0.14
Training (avg)	T	−12.46	1.66	8.03	0.66
Unseen (avg)	U	−12.42	1.43	8.23	0.39

[1] T: training angle; U: unseen angle.

From a computational perspective, the trained GRU model requires approximately 1.458 min for the offline training stage, which is performed only once. Once trained, the model infers the excitation parameters for any steering direction within the considered range in approximately 0.036 s, demonstrating its suitability for real-time and resource-constrained beamforming scenarios where repeated optimization would be computationally prohibitive.

From a beam pattern synthesis perspective, the reported results demonstrate that the GRU model is capable of learning the underlying relationship between radiation patterns

and their corresponding excitation parameters. Despite being evaluated on previously unseen steering directions, the model successfully reconstructs beam patterns that satisfy the fundamental design requirements in terms of main lobe alignment, sidelobe suppression, and directivity. This behavior highlights the effectiveness of the GRU-based approach for data-driven beam pattern synthesis in optimized uniform linear antenna arrays.

4. Discussion

The proposed framework combines evolutionary optimization and recurrent deep learning to address the beam pattern synthesis problem in uniform linear antenna arrays from a data-driven perspective. By formulating the synthesis task in terms of the array factor, the approach remains general and independent of a specific antenna implementation, which facilitates its extension to other linear array configurations and beamforming scenarios.

A key aspect of the proposed methodology is the use of a pre-optimization stage based on Differential Evolution to generate high-quality radiation patterns prior to network training. This stage enforces essential antenna performance criteria, such as controlled sidelobe behavior and adequate directivity, directly at the data generation level. As a result, the recurrent neural networks are trained on patterns that already satisfy meaningful electromagnetic constraints, allowing the learning process to focus on capturing the intrinsic relationship between radiation patterns and excitation parameters rather than compensating for poorly conditioned data.

Within this learning framework, recurrent neural models demonstrate clear advantages over simpler architectures when dealing with sequential radiation pattern representations. The comparative analysis shows that both LSTM and GRU networks significantly outperform a standard RNN, confirming the importance of gated mechanisms for capturing long-range dependencies in beam pattern data. Although the LSTM achieves the lowest error values overall, the GRU attains comparable performance while relying on a simpler internal structure. This balance between accuracy and architectural simplicity positions the GRU as an efficient alternative for data-driven beam pattern inference.

From a theoretical standpoint, the mathematical formulation and per–time-step computational cost analysis further clarify these differences. The GRU architecture requires fewer gated components than the LSTM, resulting in a lower number of dominant matrix–vector multiplications per time step. As derived in the computational cost analysis, the GRU entails $6(I + H)H$ FLOPs per time step, whereas the LSTM requires $8(I + H)H$. This reduction of approximately 25% in dominant operations provides a formal explanation for the improved computational efficiency of the GRU model. From a practical perspective, lower computational complexity not only reduces processing requirements but may also contribute to lower energy consumption in large-scale simulations and real-time or resource-constrained beamforming systems.

The beam pattern synthesis results obtained for previously unseen steering directions highlight the generalization capability of the proposed approach. Despite not being included in the training set, the evaluation patterns are accurately reconstructed in terms of main lobe direction, sidelobe suppression, and directivity. Minor discrepancies observed in the sidelobe regions are expected given the nonlinear nature of the synthesis problem, yet the overall performance remains consistent with the design objectives imposed during the pre-optimization stage. This behavior confirms that the GRU effectively learns a continuous mapping between radiation patterns and excitation parameters, enabling interpolation beyond the discrete steering angles used for training.

It is also worth noting that the proposed framework is not intended as a direct replacement for classical antenna synthesis methods, which typically require repeated optimization or closed-form evaluations for each new steering direction. The key practical advantage of

the proposed approach lies in its inference-time efficiency: once the offline training stage is completed, the GRU model infers excitation parameters for any steering direction within the considered range without requiring repeated optimization, as reported in Section 3.3. This operational advantage makes the proposed framework particularly attractive for scenarios where computational resources or response time are constrained.

Finally, the proposed combination of evolutionary optimization and recurrent learning reduces the need for exhaustive electromagnetic simulations across all possible beam directions. Once trained, the GRU model can infer excitation parameters for arbitrary steering angles within the considered range without requiring repeated metaheuristic optimization. While deep learning approaches inherently require sufficiently dense and representative training data, the results demonstrate that, when coupled with a structured pre-optimization strategy, recurrent models can provide accurate and computationally efficient beam pattern synthesis solutions.

Overall, these findings indicate that GRU-based modeling constitutes a promising tool for data-driven antenna synthesis, offering robustness, generalization capability, and computational efficiency.

5. Conclusions

This work presented a data-driven framework for beam pattern synthesis in optimized uniform linear antenna arrays, combining Differential Evolution–based pre-optimization with recurrent neural network modeling. The pre-optimization stage enabled the generation of high-quality radiation patterns that satisfy essential antenna performance requirements and serve as reliable training data for the learning stage.

A comparative analysis of recurrent architectures, supported by a formal mathematical formulation and per–time-step computational cost evaluation, showed that gated models significantly outperform a standard RNN in capturing the sequential structure of radiation patterns. While the LSTM achieved the lowest error values, the GRU provided comparable predictive performance with reduced architectural complexity. The derived FLOP analysis indicates that the GRU requires fewer dominant matrix–vector multiplications per time step than the LSTM, resulting in lower computational cost and potentially reduced energy consumption, which makes it particularly attractive for practical beamforming applications and resource-constrained scenarios.

The synthesis results for previously unseen steering directions demonstrate the generalization capability of the proposed approach, with the GRU accurately reproducing beam patterns that preserve main lobe alignment, sidelobe suppression, and directivity. Overall, the results confirm that the integration of evolutionary optimization and recurrent learning, together with a formal complexity assessment, constitutes an effective and computationally efficient strategy for data-driven beam pattern synthesis.

Author Contributions: Conceptualization, A.A., F.A. and E.S.-N.; methodology, A.A., F.A. and E.S.-N.; software, A.A. and F.A.; validation, U.P.-R., M.R.R. and A.G.-B.; formal analysis, A.A. and F.A.; investigation, A.A. and F.A.; resources, A.A., F.A., E.S.-N., M.R.R. and A.G.-B.; data curation, A.A. and F.A.; writing—original draft preparation, A.A. and F.A.; writing—review and editing, E.S.-N., M.R.R. and A.G.-B.; visualization, A.A., F.A. and U.P.-R.; supervision, A.A.; project administration, A.A. and E.S.-N.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors upon reasonable request.

Acknowledgments: A.A. acknowledges his participation in the "Investigadoras e Investigadores por México" program of the Secretariat of Science, Humanities, Technology, and Innovation (SECIHTI), under project No. 872. The authors (A.A., E.S.-N and U.P.R.) acknowledge their participation in project SECIHTI CBF-2025-G-851.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kumar, S. 6G Mobile Communication Networks: Key Services and Enabling Technologies. *J. ICT Stand.* **2022**, *10*, 1–10. [CrossRef]
2. Rappaport, T.S. *Wireless Communications: Principles and Practice*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
3. Godara, L. Applications of antenna arrays to mobile communications. I. Performance improvement, feasibility, and system considerations. *Proc. IEEE* **1997**, *85*, 1031–1060. [CrossRef]
4. Prado, D.R. The Generalized Intersection Approach for Electromagnetic Array Antenna Beam-Shaping Synthesis: A Review. *IEEE Access* **2022**, *10*, 87053–87068. [CrossRef]
5. Li, M.; Liu, Y.; Bao, Z.; Chen, L.; Hu, J.; Guo, Y.J. Efficient Phase-Only Dual- and Multi-Beam Pattern Synthesis With Accurate Beam Direction and Power Control Employing Partitioned Iterative FFT. *IEEE Trans. Antennas Propag.* **2023**, *71*, 3719–3724. [CrossRef]
6. Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
7. Vié, A. Qualities, Challenges and Future of Genetic Algorithms. *SSRN Electron. J.* **2020**. [CrossRef]
8. Derrac, J.; García, S.; Hui, S.; Suganthan, P.N.; Herrera, F. Analyzing convergence performance of evolutionary algorithms: A statistical approach. *Inf. Sci.* **2014**, *289*, 41–58. [CrossRef]
9. Karafotias, G.; Hoogendoorn, M.; Eiben, A.E. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Trans. Evol. Comput.* **2015**, *19*, 167–187. [CrossRef]
10. Taye, M.M. Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers* **2023**, *12*, 91. [CrossRef]
11. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]
12. Wang, Y.; Liu, L.; Wang, C. Trends in using deep learning algorithms in biomedical prediction systems. *Front. Neurosci.* **2023**, *17*, 1256351. [CrossRef]
13. Akinsolu, M.O.; Mistry, K.K.; Liu, B.; Lazaridis, P.I.; Excell, P. Machine Learning-assisted Antenna Design optimization: A Review and the State-of-the-art. In Proceedings of the 2020 14th European Conference on Antennas and Propagation (EuCAP), Copenhagen, Denmark, 15–20 March 2020; pp. 1–5. [CrossRef]
14. Martínez-Ramón, M.; Gupta, A.; Rojo-Álvarez, J.L.; Christodoulou, C. *Machine Learning Applications in Electromagnetics and Antenna Array Processing*; Artech House: Nordwwod, MA, USA, 2021.
15. El Misilmani, H.M.; Naous, T.; Al Khatib, S.K. A review on the design and optimization of antennas using machine learning algorithms and techniques. *Int. J. RF Microw. Comput.-Aided Eng.* **2020**, *30*, e22356. [CrossRef]
16. Sarker, N.; Podder, P.; Mondal, M.R.H.; Shafin, S.S.; Kamruzzaman, J. Applications of Machine Learning and Deep Learning in Antenna Design, Optimization, and Selection: A Review. *IEEE Access* **2023**, *11*, 103890–103915. [CrossRef]
17. Lalhriatpuii, R. Unlocking the Power of Machine Learning: Antenna Design Optimization in the Digital Age. In Proceedings of the 2023 International Conference on Advanced Computing & Communication Technologies (ICACCTech), Banur, India, 23–24 December 2023. [CrossRef]
18. Massa, A.; Marcantonio, D.; Chen, X.; Li, M.; Salucci, M. DNNs as Applied to Electromagnetics, Antennas, and Propagation—A Review. *IEEE Antennas Wirel. Propag. Lett.* **2019**, *18*, 2225–2229. [CrossRef]
19. Merad, L.; Bendimerad, F.T.; Meriah, S.M.; Djennas, S.A. Neural Networks for Synthesis and Optimization of Antenna Arrays. *Radioengineering* **2007**, *16*, 23–30.
20. Kim, J.H.; Choi, S.W. A Deep Learning-Based Approach for Radiation Pattern Synthesis of an Array Antenna. *IEEE Access* **2020**, *8*, 226059–226063. [CrossRef]
21. Kim, J.; Hong, S.-J.; Adachi, K.; Zhao, J.; Jeong, E.-R.; Joung, J. Transmit Antenna Selection Using Gated Recurrent Units for UAV-to-LEO Satellite Communications. *IEEE Trans. Veh. Technol.* **2025**, *74*, 16459–16464. [CrossRef]

22. Tarafder, P.; Ahmed, I.; Rawat, D.B.; Annavajjala, R.; Mishra, K.V. Deep Learning Model-Based Channel Estimation for THz Band Massive MIMO with RF Impairments. In Proceedings of the MILCOM 2024—2024 IEEE Military Communications Conference (MILCOM), Washington, DC, USA, 28 October–1 November 2024; pp. 282–287. [CrossRef]
23. Chandra Sahu, K.; Koziel, S.; Pietrenko-Dabrowska, A. High-Precision Antenna Modeling Using Recurrent Neural Networks With Bidirectional LSTM Layers and Dimensionality Reduction. *IEEE Trans. Antennas Propag.* **2025**, *73*, 4011–4025. [CrossRef]
24. Mallioras, I.; Zaharis, Z.D.; Lazaridis, P.I.; Pantelopoulos, S. A Novel Realistic Approach of Adaptive Beamforming Based on Deep Neural Networks. *IEEE Trans. Antennas Propag.* **2022**, *70*, 8833–8848. [CrossRef]
25. Mallioras, I.; Yioultsis, T.V.; Kantartzis, N.V.; Lazaridis, P.I.; Zaharis, Z.D. Enhancing Adaptive Beamforming in 3-D Space Through Self-Improving Neural Network Techniques. *IEEE Open J. Commun. Soc.* **2024**, *5*, 1340–1357. [CrossRef]
26. Balanis, C.A. *Antenna Theory: Analysis and Design*, 3rd ed.; John Wiley & Sons: New York, NY, USA, 2005.
27. Salem, F.M. Gated RNN: The Gated Recurrent Unit (GRU) RNN. In *Recurrent Neural Networks: From Simple to Gated Architectures*; Springer: Cham, Switzerland, 2022; pp. 85–100. [CrossRef]
28. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734. [CrossRef]
29. Sheferaw, G.K.; Mwangi, W.; Kimwele, M.; Mamuye, A. Gated Recurrent Unit Predictor Model-Based Adaptive Differential Pulse Code Modulation Speech Decoder. *EURASIP J. Audio Speech Music Proc.* **2024**, *2024*, 6. [CrossRef]
30. Cahuantzi, R.; Chen, X.; Güttel, S. A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences. In Proceedings of the Computing Conference 2023 (SAI 2023), London, UK, 22–23 June 2023; pp. 771–785. [CrossRef]
31. Lawi, A.; Mesra, H.; Amir, S. Implementation of Long Short-Term Memory and Gated Recurrent Units on Grouped Time-Series Data to Predict Stock Prices Accurately. *J. Big Data* **2022**, *9*, 89. [CrossRef]
32. Goodfellow, I.J.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 15 December 2025).
33. Drori, I. *The Science of Deep Learning*, 1st ed.; Cambridge University Press: Cambridge, UK, 2022. Available online: http://www.dlbook.org (accessed on 15 December 2025).
34. Tsantekidis, A.; Passalis, N.; Tefas, A. Chapter 5—Recurrent Neural Networks. In *Deep Learning for Robot Perception and Cognition*; Iosifidis, A., Tefas, A., Eds.; Academic Press: Cambridge, MA, USA, 2022; pp. 101–115. [CrossRef]
35. Aggarwal, C.C. *Neural Networks and Deep Learning: A Textbook*, 2nd ed.; Springer: Cham, Switzerland, 2023.
36. Salem, F.M. Recurrent Neural Networks (RNN). In *Recurrent Neural Networks: From Simple to Gated Architectures*; Springer: Cham, Switzerland, 2022; pp. 43–67. [CrossRef]
37. Chung, J.; Gülçehre, Ç.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555. [CrossRef]
38. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
39. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.