American Journal of Neural Networks and Applications
2025, Vol. 11, No. 2, pp. 88–96
https://doi.org/10.11648/j.ajnna.20251102.15

Research Article

Influence of Neural Network Learning Algorithms on High Power Amplifier (HPA) Predistortion Performance

Hariony Bienvenu Rakotonirina*, Marie Emile Randrianandrasana

Department of Telecommunication, High School Polytechnic of Antsirabe, Antsirabe, Madagascar

Abstract

In this article, we propose a feedforward neural network model designed to approximate the inverse transfer characteristic of a High-Power Amplifier (HPA) in order to linearize it using Digital Predistortion (DPD). This approach is particularly relevant for next-generation communication systems, such as those employing OTFS (Orthogonal Time Frequency Space) modulation envisioned for 6G, whose signals exhibit large amplitude variations that exacerbate amplifier nonlinearities. The performance of predistortion heavily depends on the learning algorithm used to train the neural model. We compared three optimization algorithms: Gradient Descent, Gauss-Newton, and Levenberg-Marquardt. The amplifier is modeled using the Rapp model. The neural network architecture consists of a single input neuron, a hidden layer with ten neurons using the hyperbolic tangent activation function, and a linear output neuron. Training and simulations were carried out in MATLAB, and the performance of each algorithm was evaluated using the Mean Squared Error (MSE) criterion, which quantifies the deviation between the ideal transfer characteristic of a linear amplifier and the characteristic obtained after predistortion. The results clearly show that the Levenberg-Marquardt algorithm provides the best approximation of the predistortion function, achieving an MSE on the order of 4.2708×10^{-8}, significantly outperforming Gauss-Newton 1.0481×10^{-4} and Gradient Descent (0.0272). This superior performance is attributed to Levenberg-Marquardt's ability to combine the robustness of Gradient Descent with the fast convergence of Gauss-Newton, while avoiding local minimum and issues related to poor synaptic weight initialization.

Keywords

Predistortion, Neural Network, Training Algorithm, Amplifier, Approximation

1. Introduction

Due to its performance, OTFS modulation is a potential candidate for the modulation scheme used by the 6G network. However, it creates a nonlinearity problem in the amplifier due to the high amplitude variation of the modulated signal. This problem results in a degradation of the BER (Bit Error Rate) at the receiver and an increase in the amplifier's energy consumption. To solve this problem, we proposed a neural network-based predistortion using a feedforward architecture. Figure 1 shows the overall principle of neural network-based predistortion. It involves approximating the amplifier's inverse transfer characteristic with a neural model and cascading this model with the amplifier to achieve a linear transfer characteristic [1-4]. Our contributions in this article are as follows: linearization of the amplifier's transfer characteristic using a neural model, and investigation of the influence of this neural model's learning algorithms on predistortion performance.

*Corresponding author: dao.rakotonirina@gmail.com (Hariony Bienvenu Rakotonirina)

Received: 20 October 2025; **Accepted:** 3 November 2025; **Published:** 9 December 2025

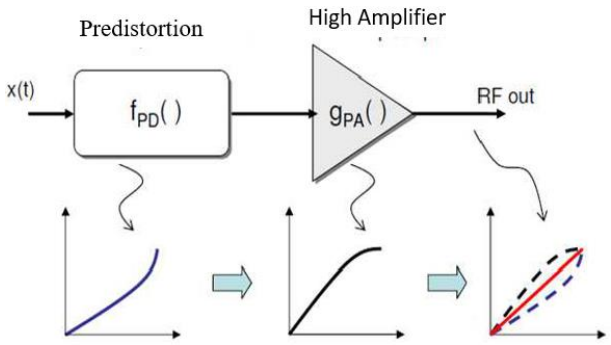

Figure 1. Principle of the predistortion technique.

2. Artificial Neural Network (ANN)

2.1. Artificial Neural

The artificial neural is a mathematical model of the operating principles of the biological neural. It receives input variables from other neurons. Each input is associated with a weight w_i, which represents the strength of the connection. The information thus gathered is processed by an activation function to produce the neural's output [5-10].

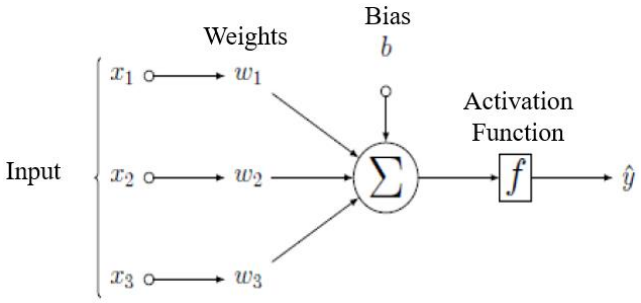

Figure 2. Artificial neural.

The inputs of the neural are represented by the column matrix x, defined by Equation (1).

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{1}$$

The matrix w represents the corresponding weights.

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \tag{2}$$

The output of the summation unit, denoted as $s(x)$, is given by the expression:

$$s(x) = \sum_{i=1}^{n} w_i x_i + b = w^T x + b \tag{3}$$

Therefore, the output of the neuron, denoted as y, is given by the expression:

$$y(x) = f\big(s(x)\big) = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4}$$

Where f is the activation function of the neuron and b is the bias (a synaptic weight corresponding to an input equal to +1).

Without an activation function a neural network would reduce to a simple linear transformation equivalent to a single neuron. The nonlinearity introduced by the activation function is therefore essential for the network to model complex, nonlinear phenomena. In the context of our paper, the activation function enables the neural model to approximate the nonlinear inverse characteristic of the power amplifier.

2.2. Feedforward Neural Network

An artificial neural network is the interconnection of multiple formal neurals. A feedforward network is a network in which all neurals in one layer are connected to all neurals in the next layer. It is the simplest type of artificial neural network, where information travels in only one direction from the input layer, through any hidden layers, to the output layer without any feedback loops. This structure is ideal for tasks like pattern recognition, image classification, and function approximation.

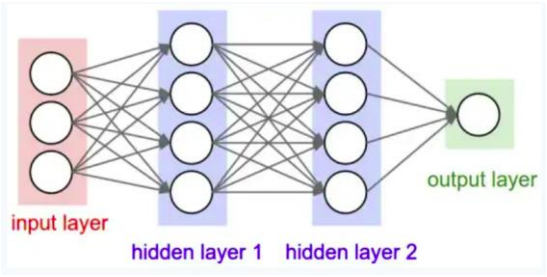

Figure 3. Feedforward Neural Network.

2.3. Neural Network Training

We are going to talk about two types of learning: supervised learning and unsupervised learning.

2.3.1. Unsupervised Learning

In the case of unsupervised learning, only the input values are available. The internal parameters of the network are adjusted using only these values; no desired outputs are taken into consideration. The network is said to be self-taught. Unsupervised learning aims to identify hidden patterns and relationships within the data, without any supervision or prior knowledge of the outcomes [11, 12].

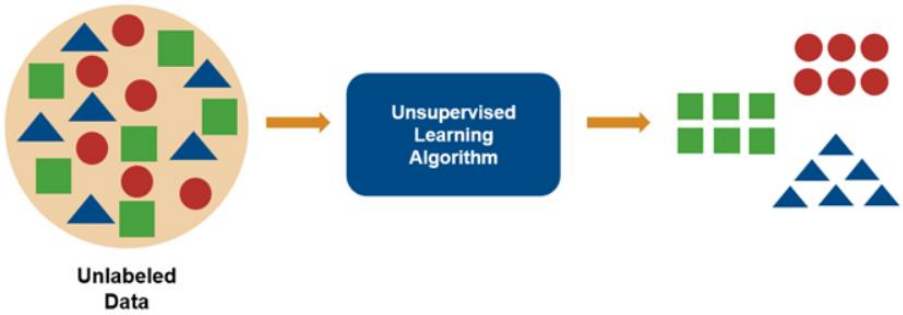

Figure 4. Unsupervised learning.

2.3.2. Supervised Learning

In supervised learning, the inputs to the network and the desired outputs are known in advance. The learning phase therefore consists of comparing the output estimated by the network with the desired output and updating the synaptic weights (w_i) until the network produces an acceptable result [13-16].

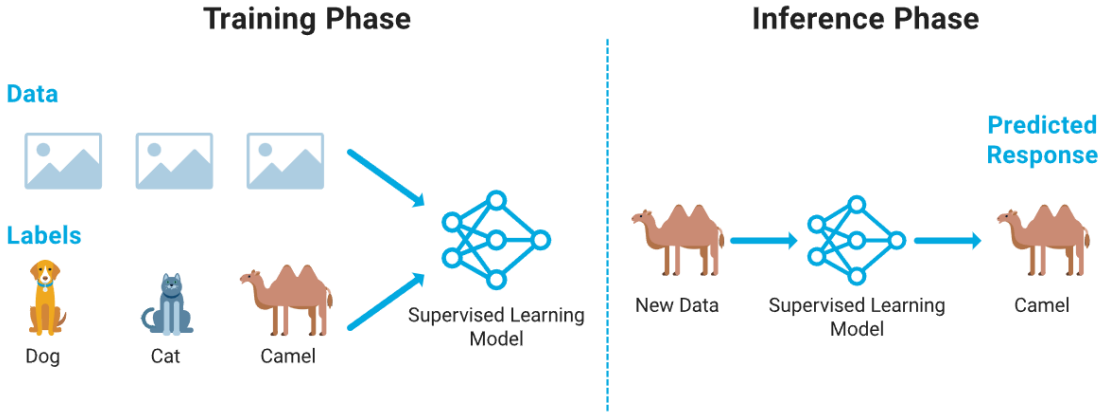

Figure 5. Supervised learning.

2.3.3. Backpropagation Algorithm

This algorithm is used for training FeedForward neural networks. Specifically, the output produced by the network is compared to the desired output, resulting in an error. This error is used to calculate its gradient, which is then propagated from the output layer to the input layer, hence the term backpropagation. This enables the adjustment of the network's synaptic weights. The operation is repeated for each input vector until the stopping criterion is met.

Here is how the algorithm works. Consider a multilayer neural network consisting of: an input layer of n_0 neurons, a hidden layer of n_1 neurons with a sigmoid activation function, and an output layer of n_2 neurons with a linear activation function.

Propagation

After initialising the synaptic weights, the network outputs are calculated by propagating the input values layer by layer, as illustrated in Figure 6.

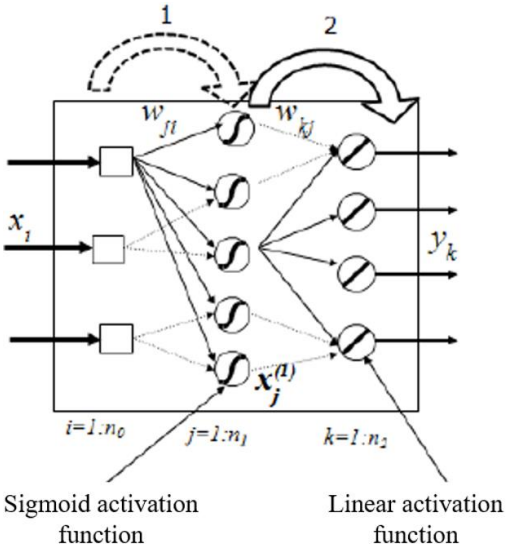

Figure 6. Propagation of input data.

Step 1

The value of the output from each neuron in the hidden layer, denoted as $x_j^{(1)}$ $where$ $j = 1, \dots, n_1$, depends on the sum of the inputs x_i $where$ $i = 1, \dots, n_0$, weighted by the synaptic weights between the input layer and the hidden layer, denoted as W_{ji}. Here, $g()$ is the activation function of the neurons in the hidden layer.

$$\begin{cases} a_j^{(1)} = \sum_{i=1}^{n_0} W_{ji} x_i \\ x_j^{(1)} = g(a_j^{(1)}) \end{cases} \quad (5)$$

Step 2

Similarly, the output value of each neuron in the output layer, denoted y_k $where$ $k = 1, \dots, n_2$., is calculated. It depends on the sum of the outputs from the hidden layer, calculated in step 1 ($x_j^{(1)}$), weighted by the synaptic weights between the hidden and output layers, denoted W_{kj}.

$$\begin{cases} a_k^{(2)} = \sum_{j=1}^{n_1} W_{kj} x_j^{(1)} \\ y_k = f(a_k^{(2)}) \end{cases} \quad (6)$$

Where $f()$ is the activation function of the neurons in the output layer.

Cost function

Let us denote by $y^{des} = [y_1^{des}, y_2^{des}, \dots, y_{n_2}^{des}]$ the desired outputs associated with the inputs x_i of neural network. The output vector of the neural network is computed by applying steps 1 and 2, resulting in $y = [y_1, y_2, \dots, y_{n_2}]$. The error between the network's actual output and the desired output is then calculated using equation (7).

$$e_k = y_k^{des} - y_k \quad (7)$$

Here is the associated cost function

$$J = \frac{1}{2} \sum_{k=1}^{n_2} e_k^2 \quad (8)$$

Backpropagation

This involves updating the synaptic weights W_{kj} and W_{ji} of the network using equation (9).

$$W_{nouvelle} = W_{ancienne} - \Delta W \quad (9)$$

If the optimization method used to minimize the cost function J is Gradient Descent, then

$$\Delta W = \lambda \frac{\partial J}{\partial W} \quad (10)$$

Where $\frac{\partial J}{\partial W}$ is the gradient of the cost function with respect to the synaptic weights of the neural network, and λ is the learning rate.

If the optimization method used to minimize the cost function J is Gauss-Newton, then

$$\Delta W = \lambda \ [J''(W)]^{-1} J'(W) \quad (11)$$

Where $J'(W) = \frac{\partial J}{\partial W}$ is the gradient of the cost function with respect to the synaptic weights of the neural network, $J''(W) = \left(\frac{\partial^2 J}{\partial W^2}\right)$ is the Hessian matrix of the cost function with respect to the synaptic weights, and λ is the learning rate.

If the optimization method used to minimize the cost function J is Levenberg-Marquardt, then

$$\Delta W = \ [J''(W) + \lambda I]^{-1} J'(W) \quad (12)$$

Let us then compute the gradient and the Hessian matrix of the cost function with respect to the synaptic weights of the neural network.

Calculation of $\frac{\partial J}{\partial W_{kj}}$ (with respect to the second layer)

Calculation of the gradient of the cost function with respect to the synaptic weights W_{kj} between the hidden layer and the output layer. This gradient is expressed as:

$$\frac{\partial J}{\partial W_{kj}} = -\sum_{k=1}^{n_2} (y_k^{des} - y_k) . f'(a_k^{(2)}) . x_j^{(1)} \quad (13)$$

Calculation of $\frac{\partial J}{\partial W_{ji}}$ (with respect to the first layer)

Calculation of the gradient of the cost function with respect to the synaptic weights W_{ji} between the hidden layer and the input layer. This gradient is expressed as:

$$\frac{\partial J}{\partial W_{ji}} = \sum_{k=1}^{n_2} -(y_k^{des} - y_k) . f'(a_k^{(2)}) . W_{kj} . g'(a_j^{(1)}) . x_i \quad (14)$$

Calculation of $\frac{\partial^2 J}{\partial W_{kj}{}^2}$ (with respect to the second layer)

Calculation of Hessian matrix of the cost function with respect to the synaptic weights W_{kj} between the hidden layer and the output layer.

$$\frac{\partial^2 J}{\partial W_{kj}{}^2} = -\sum_{k=1}^{n_2} (y_k^{des} - 1) . [f'(a_k^{(2)})]^2 . (x_j^{(1)})^2 \quad (15)$$

Calculation of $\frac{\partial^2 J}{\partial W_{ji}{}^2}$ (with respect to the first layer)

Calculation of Hessian matrix of the cost function with respect to the synaptic weights W_{ji} between the hidden layer and the input layer.

$$\frac{\partial^2 J}{\partial W_{ji}{}^2} = \sum_{k=1}^{n_2} \left(\left(\sum_{k=1}^{n_2} -(y_k^{des} - y_k) . f''(a_k^{(2)}) . W_{kj} . g'(a_j^{(1)}) . x_i \right) \ W_{kj} \right) \ g'(a_j^{(1)}) x_i \quad (16)$$

2.4. Universal Approximation Property of Neural Networks

This property is stated as follows: "for any function, there exists at least one feedforward neural network, possessing one hidden layer and a linear output neuron, that achieves an approximation of this function and its successive derivatives" [5-8]. Therefore, feedforward neural networks can be entirely suitable for modeling the desired predistortion function in order to linearize the amplifier.

3. Principle of Artificial Neural Network-Based Predistortion

The universal approximation property of neural networks implies that the predistortion function f_{PD} used to invert the amplifier's transfer characteristic, can be approximated using a non-recurrent feedforward neural network. To achieve this, the network must be trained. Therefore, a supervised learning algorithm known as the backpropagation algorithm will be employed. The following steps outline the implementation of neural network-based predistortion (see also Figure 7).

Step 1: Initialization of the synaptic weights of the feed-forward network.

Step 2: Introduction of a training dataset. Specifically, the network receives at its input the output data from the power amplifier.

Step 3: The neural network estimates the output data corresponding to the input data (i.e., the power amplifier's output data).

Step 4: Comparison between the outputs estimated by the neural network and the desired outputs, which are the input data of the power amplifier. The error between these values is defined by Equation (17). Often, this error is associated with a cost function J given by Equation (18).

$$e = y_{des} - y_{est} \tag{17}$$

With

y_{des}: desired outputs

y_{est}: outputs estimated by the neural network

$$J = \frac{1}{2}\sum(e)^2 \tag{18}$$

Step 5: Updating the synaptic weights of the neural network using Equation (19).

$$W_{nouvelle} = W_{ancienne} - \Delta W \tag{19}$$

With

$W_{ancienne}$: previous synaptic weight value

$W_{nouvelle}$: updated synaptic weight value

ΔW: correction term, which depends on the optimization algorithm used to minimize the cost function J (see Equations (10), (11), and (12)).

Step 6: The trained network is copied into the predistortion block.

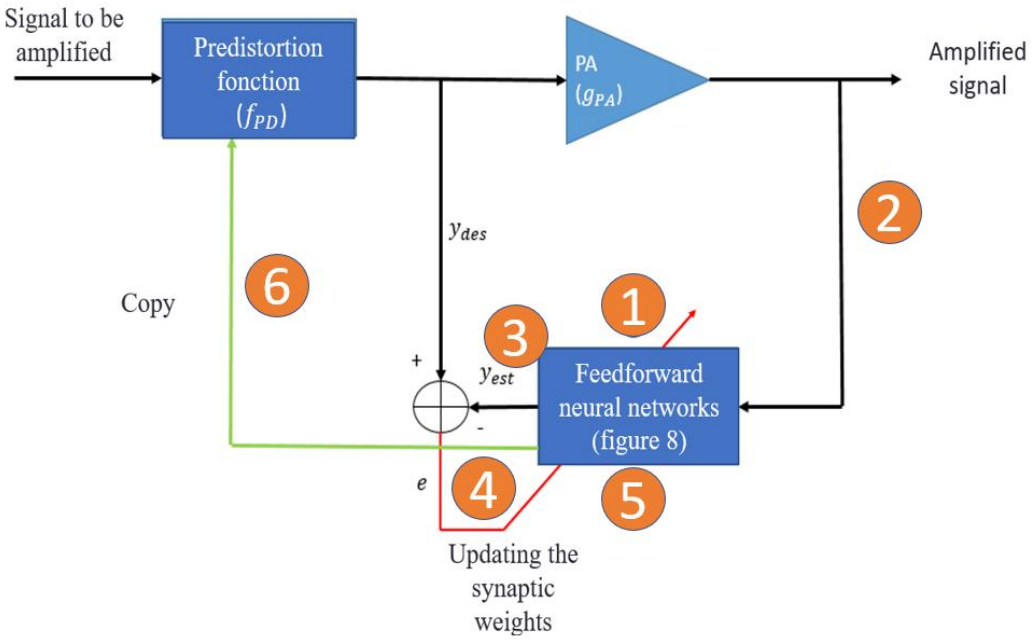

Figure 7. *Principle of neural network-based predistortion.*

4. Characteristics of the Proposed Neural Model

In this paper, we consider a terrestrial radio transmission context. For this reason, the amplifier model employed is the Rapp model. Consequently, the distortions introduced by this model affect only the amplitude of the amplified signal. Therefore, in our case, the feedforward neural network used has a single input and a single output. The accuracy of the resulting predistortion function depends on the characteristics of the neural network employed. Specifically, our feedforward network consists of three layers:

An input layer comprising a single neuron. Indeed, since the neural network has only one input, a single neuron in the input layer is appropriate.

A hidden layer with ten neurons using the hyperbolic tangent activation function. This activation function allows us to randomly initialize the synaptic weights during training, thereby accelerating the convergence of the learning algorithm.

An output layer consisting of a single neuron with a linear activation function.

In this study, we employed three high-performance learning algorithms:

Gradient descent,

Gauss-Newton,

Levenberg-Marquardt.

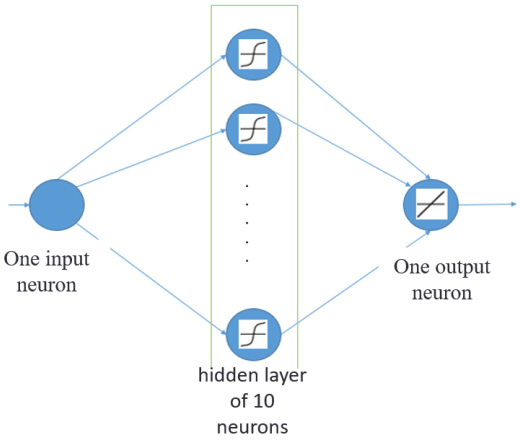

Figure 8. *Architecture of the neural network used.*

From this architecture, we derive the mathematical model of the predistortion function f_{PD} which is expressed as follows:

$$f_{PD}(x) = \sum_{i=1}^{10}\left[w_i^{cs}[g(w_i^{ec}x + b_i^c)]\right] + b_s \qquad (20)$$

w_i^{cs} is the synaptic weight between the i-th neuron of the hidden layer and the output layer neuron,

w_i^{ec} is the synaptic weight between the input layer neuron and the i-th neuron of the hidden layer,

b_i^c is the bias of the i-th neuron in the hidden layer,

b_s is the bias of the output neuron,

x is the input signal to the predistortion block,

g is the activation function of the hidden layer neurons; in our case, it is the hyperbolic tangent function defined by $g(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$

5. Performance Criterion of the Proposed Neural Model

To evaluate the performance of the proposed neural model, we used the Mean Squared Error (MSE). In our case, it represents the arithmetic mean of the squared differences between the ideal transfer characteristic of a linear amplifier (purple curve in Figures 9, 10, and 11) and the transfer characteristic obtained after linearizing a nonlinear amplifier using the neural network-based predistortion technique (black curve in Figures 9, 10, and 11).

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_i[k] - y_{est}[k])^2 \qquad (21)$$

y_i: ideal transfer characteristic of a linear amplifier.

y_{est}: transfer characteristic obtained after linearizing a nonlinear amplifier using the neural network-based predistortion technique.

6. Results

The experiment was conducted using MATLAB 2023 installed on a machine with the following specifications: CPU: Intel(R) Core i7-9750H, GPU: NVIDIA GetForce RTX 2060, RAM: 16Go.

Recall that the power amplifier model used is the RAPP model [17]. The predistortion function is modeled using the neural network described in Section 4. Three methods are employed to minimize the cost function: Gradient Descent, Gauss-Newton and Levenberg-Marquardt.

Figures 9, 10, and 11 show the shape of the predistortion function obtained from the proposed neural model. These figures also display the amplifier's transfer characteristic after neural network-based predistortion. And Table 1 provides the MSE values.

Table 1. *MSE values for each learning algorithm.*

	Descente du gradient	Gauss-Newton	Levenberg Marquardt
MSE	0.0272	1.0481×10^{-4}	4.2708×10^{-8}

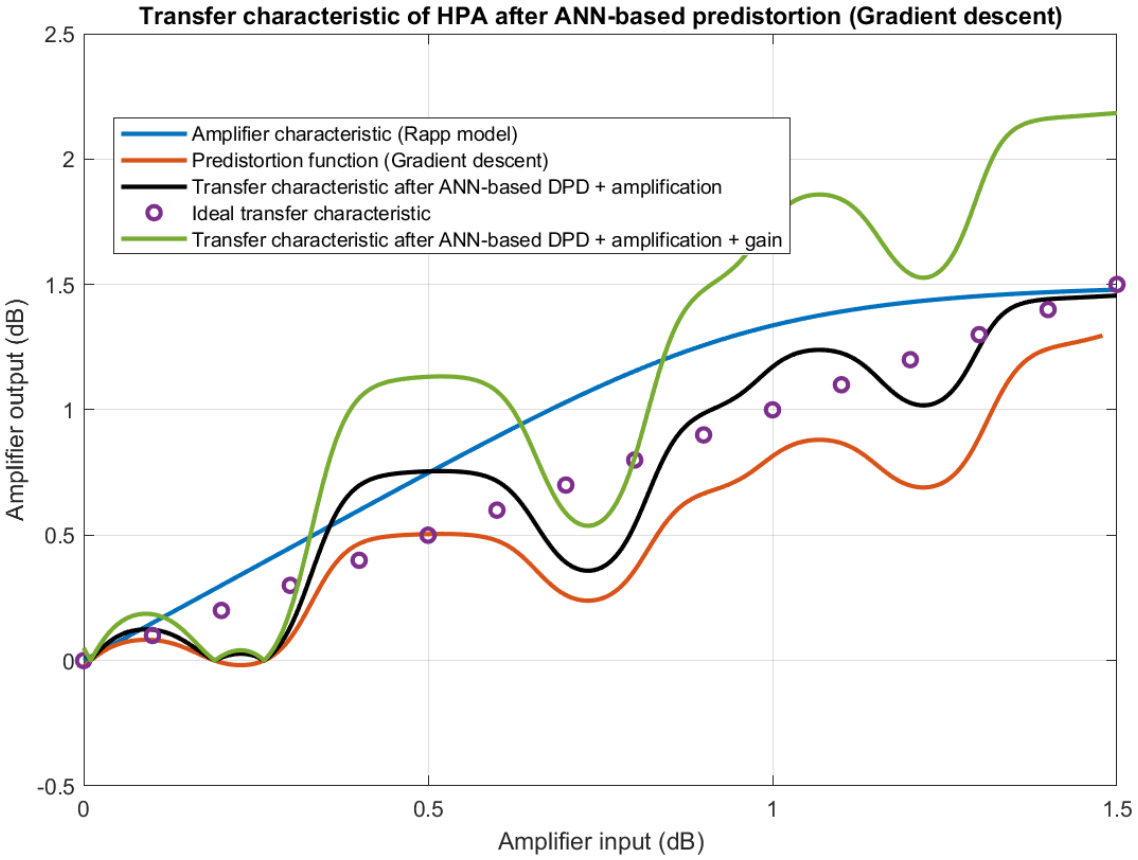

Figure 9. *Performance of Neural Network-Based Predistortion Using Gradient Descent.*

Figure 10. *Performance of Neural Network-Based Predistortion Using Gauss-Newton.*

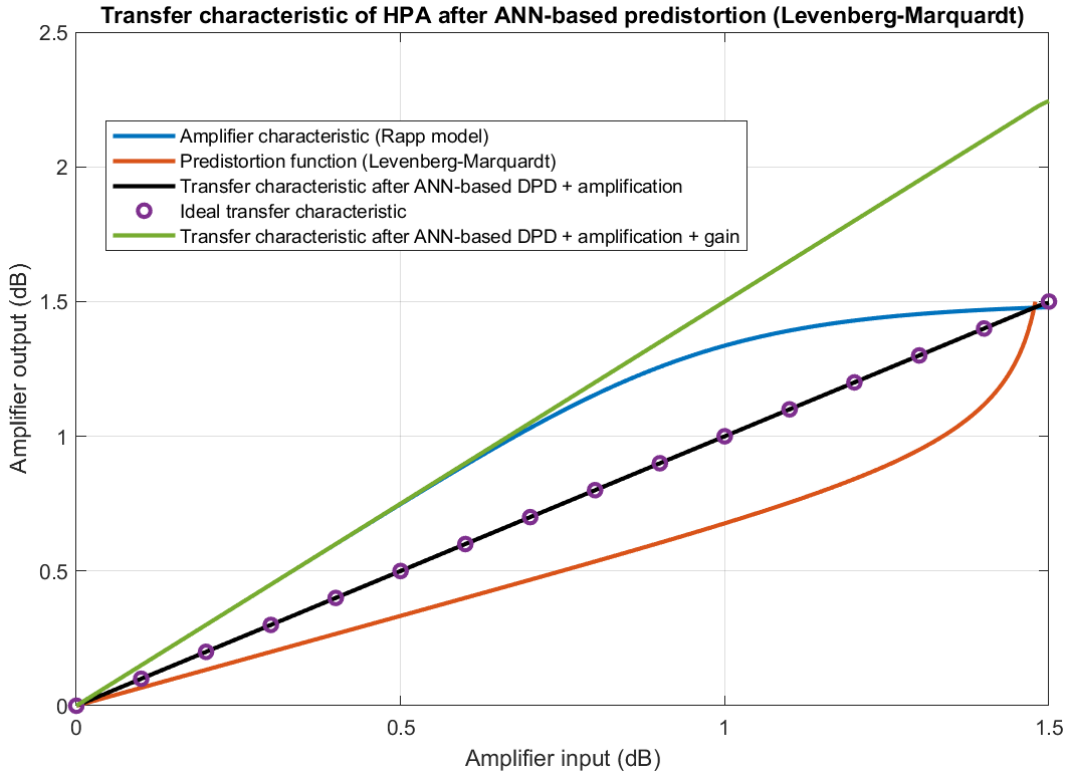

Figure 11. *Performance of Neural Network-Based Predistortion Using Levenberg-Marquardt.*

The goal of predistortion is to compensate for the amplifier's nonlinearity (Rapp model, blue curve) so that the overall chain (predistortion function + amplifier) exhibits a transfer characteristic as close as possible to the ideal straight line (purple markers), i.e., a linear response with constant gain.

In Figure 9, the transfer characteristic of the amplifier after predistortion and amplification (black curve) deviates significantly from the ideal characteristic (purple points). This indicates that the neural network failed to accurately model the predistortion function (brown curve). Indeed, gradient descent is a simple but slow algorithm that can easily get stuck in local minima and fail to converge properly toward the global minimum.

In Figure 10, the transfer characteristic after predistortion and amplification (black curve) follows the ideal trajectory (purple points) much more closely than in the previous case. The deviations are reduced, as reflected by the lower MSE value. The Gauss-Newton algorithm uses a quadratic approximation of the cost function which accelerates convergence compared to gradient descent. However, it can become unstable if the synaptic weights are poorly initialized or if the approximated Hessian matrix is ill-conditioned.

In Figure 11, the transfer characteristic after predistortion and amplification (black curve) follows the purple points of the ideal characteristic almost perfectly. The deviation is minimal, as confirmed by the low MSE value. The overall response (green curve) is nearly linear, indicating that the neural network was able to accurately model the inverse of the amplifier's nonlinearity (brown curve). The Levenberg-Marquardt algorithm combines the advantages of gradient descent (stability) and Gauss-Newton (speed). It dynamically adjusts a regularization parameter to ensure convergence even when the synaptic weight initialization is poor.

7. Conclusion

In this article, we investigated the influence of learning algorithms on the performance of digital predistortion based on a feedforward neural network applied to a power amplifier modeled using the Rapp model. The objective was to approximate the inverse transfer characteristic of the amplifier in order to compensate its nonlinearities, which are particularly critical in 6G communication systems employing high-PAPR (Peak-to-Average Power Ratio) modulation schemes such as OTFS. Three optimization algorithms were compared: Gradient Descent, Gauss-Newton, and Levenberg-Marquardt. The results clearly show that the Levenberg-Marquardt algorithm achieves the highest accuracy. This superior performance stems from its ability to combine the robustness of Gradient Descent with the fast convergence of the Gauss-Newton method. The logical next step in our research is to apply the proposed neural network–based predistortion technique within a telecommunications system such as 6G to compensate for amplifier nonlinearity, thereby reducing transmission errors and the system's energy consumption. These findings confirm that the choice of learning algorithm is a decisive factor in the effectiveness of neural-network-based linearization. The proposed approach thus opens promising prospects for integrating adaptive lineariza-

tion solutions into transmitters for future 6G networks.

Abbreviations

6G	Sixth Generation of Mobile Telephony
ANN	Artificial Neural Network
BER	Bit Error Rate
DPD	Digital Predistortion
HPA	High-Power Amplifier
MATLAB	Matrix Laboratory
MSE	Mean Squared Error
OTFS	Orthogonal Time Frequency Space
PAPR	Peak-to-Average Power Ratio

Author Contributions

Hariniony Bienvenu Rakotonirina: Conceptualization, Formal Analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Writing – original draft, Writing – review & editing

Marie Emile Randrianandrasana: Supervision, Validation

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Li, W., Montoro, G., & Gilabert, P. L. (2025). Adaptive Digital Predistortion for User Equipment Power Amplifiers Under Dynamic Frequency, Bandwidth, and VSWR Variations. IEEE Transactions on Microwave Theory and Techniques, PP (99): 1-13. https://doi.org/10.1109/TMTT.2025.3563159

[2] Wu, Y., Li, A., Beikmirza, M., Singh, G. D., Chen, Q., de Vreede, L. C. N., Alavi, M., & Gao, C. (2024). MP-DPD: Low-Complexity Mixed-Precision Neural Networks for En-ergy-Efficient Digital Predistortion of Wideband Power Amplifiers, IEEE Microwave and Wireless Technology Letters, https://doi.org/10.1109/LMWT.2024.3386330

[3] Spano, C., Badini, D., Cazzella, L., & Matteucci, M. (2025). Local and Remote Digital Pre-Distortion for 5G Power Amplifiers with Safe Deep Reinforcement Learning. *Sensors*, 25(19), 6102. https://doi.org/10.3390/s25196102

[4] Thys, C., Martinez Alonso, R., Lhomel, A., Fellmann, M., Deltimple, N., Rivet, F., & Pollin, S. (2024). Walsh-domain Neural Network for Power Amplifier Behavioral Modelling and Digital Predistortion. IEEE International Symposium on Circuits and Systems (ISCAS), https://doi.org/10.1109/ISCAS58744.2024.10557970

[5] Honda, H. (2023). *Universal approximation property of a continuous neural network based on a nonlinear diffusion equation.* Advances in Continuous and Discrete Models, 2023, Springer. https://doi.org/10.1186/s13662-023-03787-z

[6] Ismailov, V. (2024). Universal approximation theorem for neural networks with inputs from a topological vector space. arXiv preprint, https://doi.org/10.48550/arXiv.2409.12913

[7] Geonho, H., Wonyeol, L., Yeachan, P., (2025). Float-ing-Point Neural Networks Are Provably Robust Universal Approximators. Arxiv, https://doi.org/10.48550/arXiv.2506.16065

[8] Vital, W. L., Vieira, G., & Valle, M. E. (2022). Extending the Universal Approximation Theorem for a Broad Class of Hypercomplex-Valued Neural Networks. *Arxiv*, 158, 563-575. https://doi.org/10.1016/j.neunet.2022.09.024

[9] Aggarwal, C. C. (2023). Neural Networks and Deep Learning: A Textbook (2e éd.). Springer Cham. https://doi.org/10.1007/978-3-031-29642-0

[10] Maharajan, C., Chandran, S., Changjin, X., Lagrange stability of inertial type neural networks: A new LKF approach, Evolving Systems 16 (2025) 63. https://doi.org/10.1007/s12530-025-09681-1

[11] Qingyi, C., Changjin, X., Bifurcation and controller design of 5D BAM neural networks with time delay, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields (2024). https://doi.org/10.1002/JNM.3316

[12] Jo, T. (2022). Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning. Springer Cham. https://doi.org/10.1007/978-3-030-65900-4

[13] Cerulli, G. (2023). Fundamentals of Supervised Machine Learning: With Applications in Python, R, and Stata. Springer Cham. https://doi.org/10.1007/978-3-031-41337-7

[14] Amini, M.-R. (2025). Advanced Supervised and Semi-supervised Learning: Theory and Algorithms. Springer Cham. https://doi.org/10.1007/978-3-031-99928-4

[15] Qingyi, C., Changjin, X, Further study on Hopf bifurcation and hybrid control strategy in BAM neural networks concerning time delay, AIMS Mathematics 9(5) (2024) 13265–13290, https://doi.org/10.3934/math.2024647

[16] Maharajan, C., Chandran, S., Changjin, X., Delay dependent complex-valued bidirectional associative memory neural networks with stochastic and impulsive effects: an exponential stability approach, Kybernetika 60(3) (2024) 317–356, https://doi.org/10.14736/kyb-2024-3-0317

[17] Kostrzewska, K., & Kryszkiewicz, P. (2024). Power Amplifier Modeling Framework for Front-End-Aware Next-Generation Wireless Networks. Electronics, Mdpi. https://doi.org/10.3390/electronics13091643

Research Field

Hariniony Bienvenu Rakotonirina: Telecommunication, Signal processing, Compressive Sensing, Artificial intelligence, High Amplifier Power Nonlinearity

Marie Emile Randrianandrasana: Telecommunication, Signal processing, Compressive Sensing, Radar, Electromagnetic wave