DOI: https://doi.org/10.15407/rpra31.02.098
UDC 004.8:621.396.67

S. Butov[1], A. Tuz[2], O. Morozova[2], S. Vinnichenko[1], O. Khardikov[1], S. Shulga[1]

[1] School of Radiophysics, Biomedical Electronics and Computer System
V.N. Karazin Kharkiv National University
4, Svobody Square, Kharkiv, 61022, Ukraine

[2] Department of Computer Systems, Networks and Cybersecurity
of National Aerospace University "KhAI"
17, Vadym Manko St., Kharkiv, 61070, Ukraine
E-mail: nnc.met50@i.ua

COMPUTATIONAL EFFICIENCY OF AN ARTIFICIAL NEURAL NETWORK IN OPTIMIZING THE OPERATING CONDITIONS OF PATCH ANTENNAS

Subject and Purpose. *Designing antennas with desired operational features is a complex optimization problem addressing a large number of parameters and nonlinear relationships. Artificial neural networks (ANNs) have had strong potential in antenna engineering. The ability to approximate nonlinear functions and capture hidden relationships enables partial automation of antenna designing. However, it requires a detailed analysis of how ANN parameters affect predictive accuracy and computational efficiency. The present study investigates the influence of ANN architecture and training configurations on the predictive accuracy of resonant frequencies for rectangular microstrip patch antennas.*

Methods and Methodology. *A modular Python-based experimental platform is used to evaluate ANN performance for different ANN configurations. The ANN training is on a synthetic dataset generated from a classical analytical antenna model. The number of hidden layers, neurons per layer, activation functions, and optimizers are systematically varied to assess their particular impacts on convergence, generalization performance, and execution time.*

Results. *It has been shown that a three-layer neural network [1024, 512, 256 neurons] with ReLU activation and the Adam optimizer strikes the best balance between predictive accuracy and training rate. Simpler or excessively deep architectures, non-adaptive optimizers, and saturating activation functions can slow convergence or cause unstable training. Further analysis indicated that a smaller batch size introduces useful stochasticity into training but might also destabilize the process.*

Conclusions. *The study has demonstrated that joint optimization of ANN architecture and training dynamics is essential for developing accurate and computationally efficient electromagnetic models. The practical results are recommendations for machine-learning-based antenna design.*

Keywords: *artificial neural network (ANN), activation function, adaptive optimizer, microstrip patch antenna, automatic antenna design.*

Introduction

Designing antennas with desired performance is a an optimization challenge. It depends on numerous interrelated parameters, including geometry, substrate materials, constitutive parameters, and feeding techniques. Classical analytical approaches [1—3] provide accurate but computationally expensive methods that scale poorly with increasing model complexity or extensive parametric optimization.

Citation: Butov, S., Tuz, A., Morozova, O., Vinnichenko, S., Khardikov, O., Shulga, S., 2026. Computational efficiency of an artificial neural network in optimizing the operation conditions of patch antennas. *Radio Phys. Radio Astron.*, **31**(2), pp. 98—107. https://doi.org/10.15407/rpra31.02.075

Recent years have seen significant advances in applying artificial intelligence (AI) to microwave component design [4—9]. Among AI techniques, artificial neural networks (ANNs) have demonstrated remarkable potential to augment or replace traditional modeling tools in antenna engineering [10—13]. ANNs have been effective for estimating the resonant frequencies of patch antennas in terms of geometric and material parameters. They aid in the automatic design of microwave components and support simulations of complex electromagnetic structures, such as metasurfaces. The inherent ability of ANNs to approximate nonlinear functions and extract hidden relationships from data makes them particularly appealing for handling electromagnetic design problems.

Despite advances and promising prospects, the literature suggests [14] that neural network evolution focuses primarily on improving predictive accuracy. Much less attention is paid to the structural and training characteristics of neural networks themselves. ANN parameters (hidden layers, neurons per layer, activation function class, optimizer type, batch size, and training epoch count) are often selected heuristically and seldom justified in depth. Studies that do consider architectural design choices (see, e.g., [15—18]) frequently lack systematic numerical experiments intended to quantitatively assess the impacts on convergence dynamics, generalization ability, and training efficiency.

This paper seeks to address the existing gap by systematically analyzing how different ANN architectures and training configurations influence the learning behavior and predictive accuracy of the resonant frequency for rectangular microstrip antennas, a commonly used type in wireless communication systems. Using a consistent synthetic dataset derived from electromagnetic theory, we evaluate the influence of architectural and hyperparameter variations on performance, aiming to identify configurations that balance predictive accuracy, computational efficiency, and robustness.

1. Experimental Setup and Reference Model Determination

A modular experimental platform was implemented to analyze the ANN's computational efficiency in predicting resonant frequencies for rectangular microstrip antennas. The workflow covers all stages from synthetic dataset generation to training, validation, and performance assessment, thereby ensuring full control and reproducibility.

Python is appreciated for its flexibility, intuitive syntax, and rich ecosystem of scientific computing libraries, and it was selected as a primary programming language. It has become a de facto standard in machine learning and numerical modeling, offering mature frameworks such as TensorFlow, PyTorch, Keras, and Scikit-learn, enabling the seamless integration between numerical models and neural network components.

All computational experiments are run using PyCharm Professional IDE on macOS with an Apple Silicon processor. This environment offers robust support for virtual environments (venv, conda), intelligent autocompletion, visual tools for tracking training metrics, and tight integration with visualization libraries. This setup significantly improves productivity and reproducibility during computational experiments.

TensorFlow 2.x is chosen as the core machine-learning framework, utilizing its high-level Keras API to define, compile, and train neural networks. Keras' modular and user-friendly design enables rapid prototyping of network architectures and effortless tuning of hyperparameters, including the number of layers, activation functions, optimizers, batch sizes, and epoch counts. To create the training dataset, the data generator is implemented in C# language.

The generator is implemented as a classical physical model for calculating the resonant frequency f of a patch antenna (Fig. 1). It represents a conducting radiating patch of length L and width W on a dielectric substrate of thickness h and relative permittivity ε_r with a conducting ground plate underneath. The resonant frequency f_{mn} of the rectangular patch can be calculated by the formula [12]

$$f_{mn} = \frac{c}{2\sqrt{\varepsilon_{eff}}}\left[\left(\frac{m}{L_{eff}}\right)^2 + \left(\frac{n}{W_{eff}}\right)^2\right]^{1/2},$$

where ε_{eff} is the effective relative permittivity of the substrate, c is the speed of electromagnetic waves in free space, m and n are the integers defining the order of the mode, and L_{eff} and W_{eff} are the effective dimensions of the patch. The patch width directly

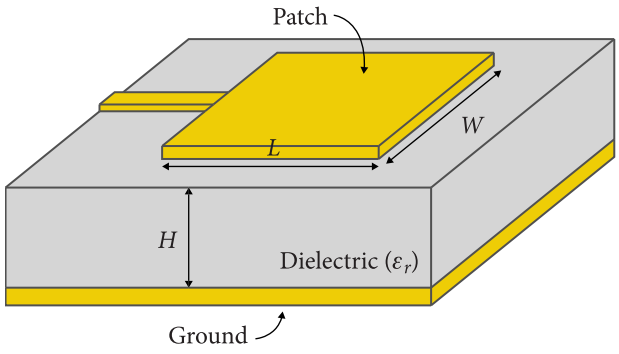


Fig. 1. The patch antenna

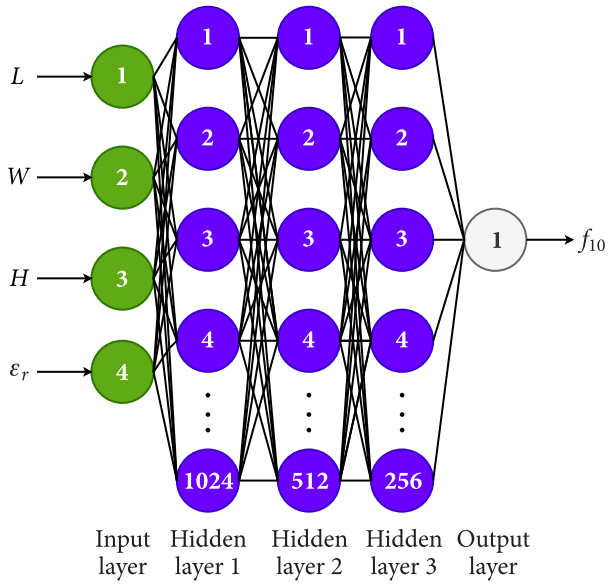


Fig. 2. The ANN architecture of the reference model

influences a series of important antenna parameters, including the radiation pattern and frequency at which the antenna is expected to resonate. The resonant frequency f_{mn} of the fundamental TM_{10} mode of the antenna is determined as

$$f_{10} = \frac{c}{2L_{eff}\sqrt{\varepsilon_{eff}}}, \quad (1)$$

where $L_{eff} = L + 2\Delta L$.

The effective dielectric constant of the substrate and the patch length extension are derived from empirical models commonly used in antenna theory [19—22]. In this study, the effective permittivity is approximated as

$$\varepsilon_{eff}(W) = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\frac{1}{\sqrt{1 + 10h/W}}, \quad (2)$$

and the length extension is

$$\Delta L = 0.412h\frac{\left(\varepsilon_{eff}(W) + 0.300\right)\left(W/h + 0.264\right)}{\left(\varepsilon_{eff}(W) - 0.258\right)\left(W/h + 0.813\right)}. \quad (3)$$

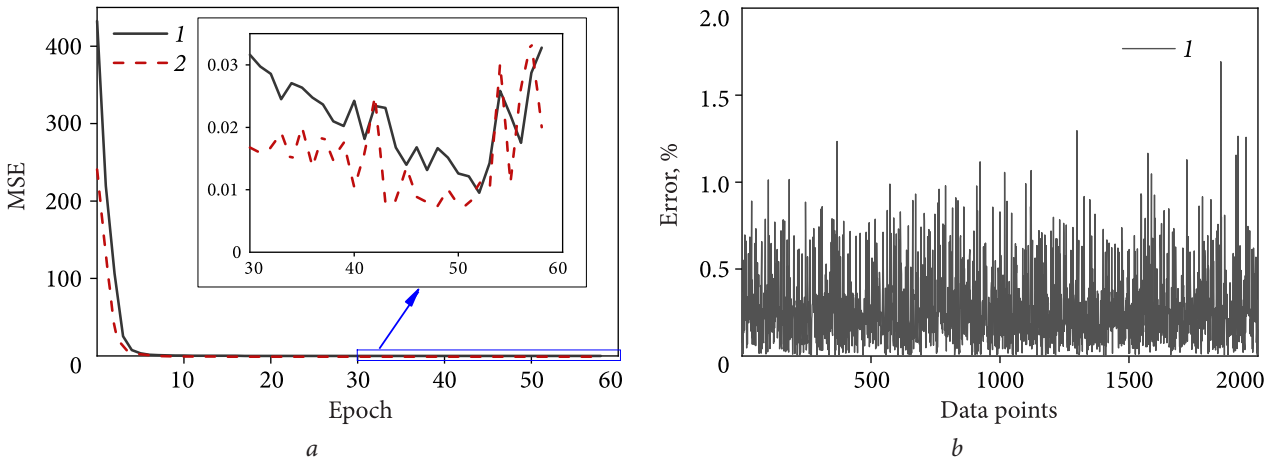
Equations (1)—(3) are the pillars of the standard closed-form description of a rectangular microstrip patch antenna operating in its fundamental mode. Obtained beyond full-wave simulations, these expressions are commonly used in antenna design as they provide are liable first-order approximation of resonant-frequency behavior. It should be emphasized that although these equations do not capture higher-order effects, they remain accurate enough to serve as a reference for generating synthetic datasets in machine learning experiments [23]. To obtain more accurately calculated antenna characteristics, it is recommended to use specialized electromagnetic simulation software, such as CST or Ansys HFSS.

The input data fed to the ANN include thickness h and relative permittivity ε_r of the substrate and effective width and length, W and L, of the antenna. The target output is the resonant frequency f_{10} obtained from Eq. (1). The 1.6 GHz operating frequency was selected because it falls within the L-band, which is extensively used in satellite navigation systems. This frequency range provides a practical balance between antenna size, propagation characteristics, and signal attenuation, making it relevant for real-world antenna applications. Also, this range provides a normalization of the influence of the antenna parameters on the ANN's resonant-frequency prediction results.

Two independent datasets, each with 2000 examples, were generated using the same antenna model and different random input values to ensure statistical independence. One dataset served as a held-out test set for final evaluation. The other was split 80/20 into training and validation subsets (validation_split = 0.2), with shuffling enabled at each epoch. During training, model performance was monitored on the validation set, which was also used for early stopping (monitor = val_loss, patience = 10, restore_best_weights = True).

2. Results and Discussion

The computational experiments show how much the changes in various ANN parameters affect the predictive accuracy and computational efficiency in rectangular-patch resonant frequency calculations.



a *b*

Fig. 3. Convergence rate (*a*) and relative error (*b*) for the frequencies predicted by the ANN-trained reference model

The reference ANN model used throughout the computational experiments is a fully connected feedforward network with three hidden layers consisting of 1024, 512, and 256 neurons, respectively (Fig. 2). All the hidden layers use the ReLU activation function, while the output layer is linear to suit the regression task [24, 25]. This configuration was selected based on the initial testing, which demonstrated both good convergence behavior and acceptable prediction error. The ANN models were trained using the Adam optimizer and the Mean Squared Error (MSE) loss. The batch size was 64 for the forward model and 128 for the inverse model, with, respectively, 100 and 300 epochs as upper bounds.

This reference ANN model serves as the checkpoint for all subsequent computational experiments in which specific parameters (number of layers, activation function, optimizer, etc.) are varied to study their specific effects. It provides a baseline for evaluating the effects of architecture and hyperparameter modifications. In particular, the model is featured by low training and validation errors (0.0100 and 0.0107, respectively), demonstrating fast and stable convergence within 11 seconds (Fig. 3, *a*). In Fig. 3, *b*, the frequencies predicted by the trained ANN are imposed on the frequencies calculated from Eq. (1) for each of the 2000 samples. The relative frequency deviations come from the formula

$$Error = \frac{\left| f_{10}^{predicted} - f_{10}^{analytic} \right|}{f_{10}^{analytic}} \cdot 100\%.$$

Importantly, because the algorithm incorporates built-in protection against overfitting, training may require significantly fewer than the 100 epochs specified in the model, as observed in the reference model (Fig. 3, *a*). Note that, hereinafter, in all convergence rate MSE plots, the odd-index curves correspond to the training error of the model under study, and the even-index curves correspond to the validation error of the same model.

Detailed configuration parameters of the reference model are summarized below:

number of layers	input layer, 3 hidden layers, output layer;
activation function	ReLU;
optimizer	Adam;
loss-function	MSE;
number of epochs	100;
batch size	128;
EarlyStopping patience	10.

2.1. Effect of Network Depth

The number of hidden layers determines the ANN depth, which plays a fundamental role in the network's capacity to model complex relationships between input features and output targets [24]. In the context of resonant frequency prediction for a rectangular patch antenna, the ANN's depth significantly affects the network capacity to approximate nonlinear dependences guided by physical laws.

To study this effect, the reference model is compared with two other network configurations. One is shallow, with two hidden layers instead of three. The other is deep, with four hidden layers (Fig. 4). All other conditions, including activation function (ReLU), optimizer (Adam), epoch count (100), and

batch size (128), remain unchanged to isolate the impact of depth.

The obtained results show that reducing the number of layers significantly deteriorates performance. The training error increases, reaching 0.0198, at least double its previous value. The validation error rises to 0.0201 (see curves *1* and *2* in Fig. 4, *a* and curve *1* in Fig. 4, *b*). Notice that the training process is stochastic because of, in particular, a random partitioning of the initial data set. Therefore, to assess the role of a specific model parameter, the ANN training was run several times with parameters fixed. Across those training runs, the training error of the two-hidden-layer model increased several times. This indicates that the network lost its capacity to sufficiently generalize and reproduce the underlying functional relationship between geometric and dielectric parameters and the resonant frequency. Too shallow, the network cannot likely acquire the needed abstraction levels, limiting the network expressiveness and convergence potential.

Conversely, adding the fourth layer moderately improves the network's generalization ability (curves *3* and *4* in Fig. 4, *a* and curve *2*, Fig. 4, *b*): in the best-case ANN, the validation error drops to 0.006. The error decreases due to a 30% extension of the training time, from 11.3 to 14.9 seconds. The training error remains relatively stable, indicating that the added layer contributes more to the generalization than to the training accuracy. Thus, although deeper networks can somewhat improve generalization, the computational cost often outweighs the benefit.

These findings align with broader observations in the literature [9, 26]. Namely, deeper networks tend to perform better. However, at a certain depth, the gains plateau or even diminish because of overfitting, vanishing gradients, or increased training instability. In this case, a three-layer architecture strikes a good balance between learning capacity and efficiency, making it the most effective option for the set goal.

2.2. Effect of Activation Function

The choice of the activation function plays a crucial role in ANN expressiveness, convergence behavior, and stability. The present study evaluates three of the most common activation functions: ReLU (Rectified Linear Unit), sigmoid, and tanh. Each of them introduces non-linearity, which is essential for the network's ability to learn complex relationships between input parameters and target output. In this study, the target output is the resonant frequency of a rectangular patch antenna.

The reference model uses the ReLU activation function, which has become a *de facto* standard in modern deep learning due to its simplicity and computational efficiency. In this study, ReLU exhibited robust training dynamics with a training error of 0.0100 and a validation error of 0.0107. Its linear behavior in a positive-valued domain allows gradients to propagate efficiently, accelerating convergence and preventing the vanishing-gradient problem [15]. Also, ReLU performs reliably across the other network architecture variants. It is robust to overfitting, and it consistently achieves high accuracy.

In contrast, the usage of the sigmoid activation function is responsible for dramatic training failures (curves *1* and *2* in Fig. 5, *a* and curve *1*, Fig. 5, *b*). At best, exploding gradients inflate training and validation errors to 174.8 and to 186.4, respectively, making these errors more than two orders of magnitude higher than those in the ReLU-based model. Besides, the convergence behavior is unstable, with NaN (Not a Number) values on the error curve. This is likely due to the saturation problem inherent in the sigmoid function, which causes gradients to become too small for large input values and prevents effective updating the weights in deep networks.

The tanh activation function performance is intermediate between the sigmoid and ReLU functions, see curves *3* and *4* in Fig. 5, *a* and curve *2* in Fig. 5, *b*. Its training and validation errors are, respectively, 0.1121 and 0.0395, which is significantly better than the sigmoid counterparts but still loses against ReLU. Though suffering from saturation at extreme values, tanh centers outputs around zero, which often aids convergence in some architectures. In the present task, however, tanh introduces slower and less stable convergence. The training convergence rate decreases significantly, indicating that although the network eventually generalizes across datasets, it takes longer and is more sensitive to hyperparameter tuning.

Overall, the computational experiments confirm that of the three, ReLU is best suited for this regression task due to its non-saturation behavior, computational efficiency, and consistent gradient flow.

The sigmoid failure demonstrates that bounded activation functions are not suitable for deeper ar-

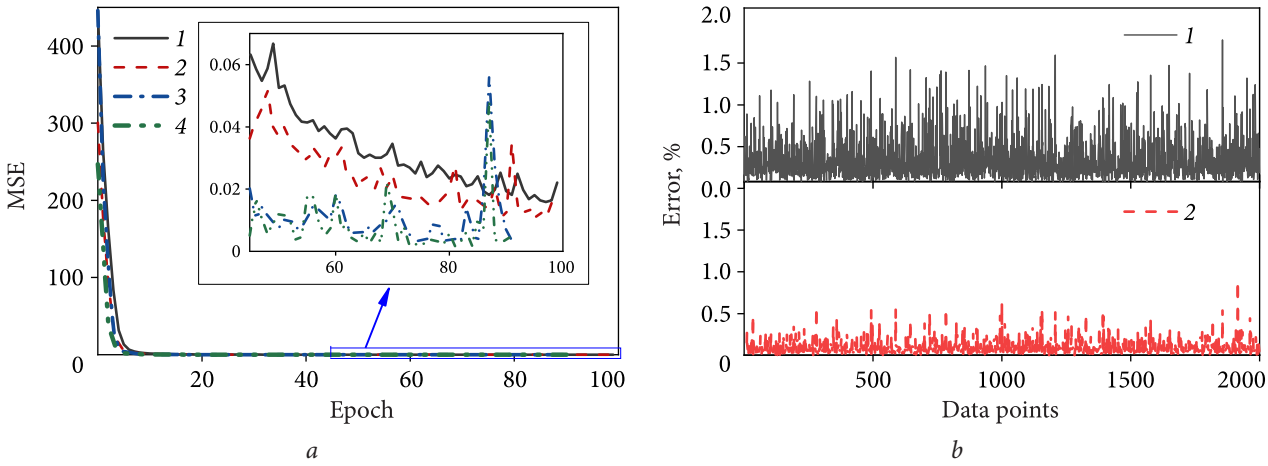


Fig. 4. Convergence rate (*a*) and relative error (*b*) for the frequencies predicted by the ANN-trained model with different numbers of hidden layers: curves *1* and *2* in (*a*) and curve *1* in (*b*) — with two hidden layers; curves *3* and *4* in (*a*) and curve *2* in (*b*) — with four

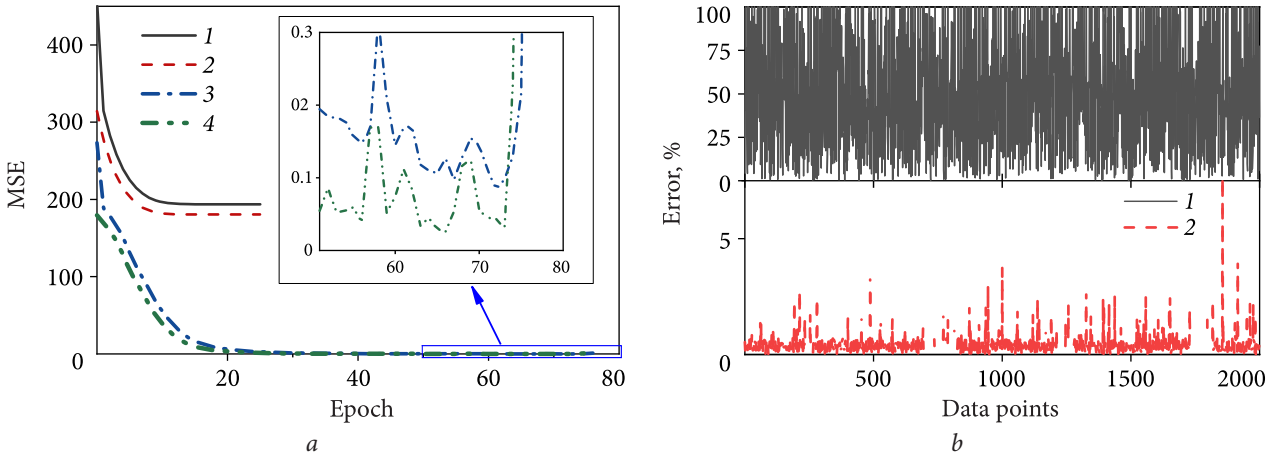


Fig. 5. Convergence rate (*a*) and relative error (*b*) for the frequencies predicted by the ANN-trained model with different activation functions: curves *1* and *2* in (*a*) and curve *1* in (*b*) — sigmoid; curves *3* and *4* in (*a*) and curve *2* in (*b*) — hyperbolic tangent (tanh)

chitectures in physical modeling. Tanh, while more stable than sigmoid, still falls short of ReLU performance in both speed and error minimization.

2.3. Effect of Optimizer Choice

The choice of optimizer plays a pivotal role in determining how effectively an ANN converges during training. Three common optimizers — Adam (Fig. 3), RMSprop, and Stochastic Gradient Descent (SGD) (Fig. 6) — were systematically evaluated for performance using the same network architectures and training conditions to isolate the effect of the optimization strategy.

The Adam optimizer strikes an ideal balance between training rate and prediction accuracy (see Fig. 3). As a part of the reference model, the Adam optimizer favors rapid and stable convergence, resulting in a training error as low as 0.0100 and a validation error as low as 0.0107 within approximately 11.3 seconds of training time. Its adaptive learning rate adjustment mechanism effectively updates weights of different layers, promoting robustness even in deeper neural networks.

In contrast, RMSprop yields much worse results, with training and validation errors as high as 7.24 and 5.36, respectively (see curves *1* and *2* in Fig. 6, *a*, and curve *1* in Fig. 6, *b*). Even though the training was completed, the convergence was not as effective as in the Adam optimizer case. This may be attributed to RMSprop's sensitivity to initialization parameters and learning rate settings. Not necessarily

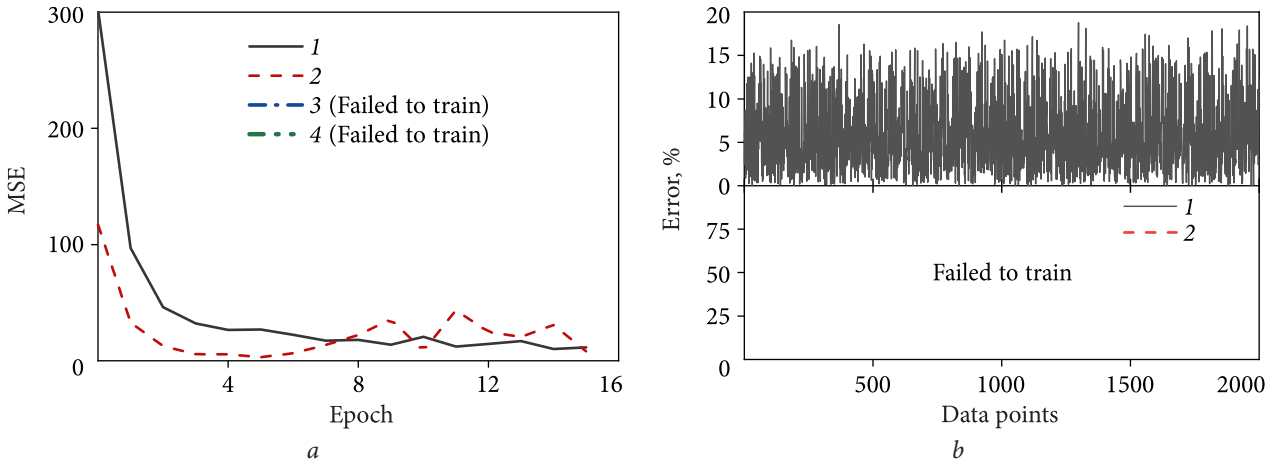


Fig. 6. Convergence rate (*a*) and relative error (*b*) for frequencies predicted by the ANN-trained model using different optimizers: curves *1* and *2* in (*a*) and curve *1* in (*b*) — RMSprop optimizer; curves *3* and *4* in (*a*) and curve *2* in (*b*) — stochastic gradient descent (SGD) optimizer

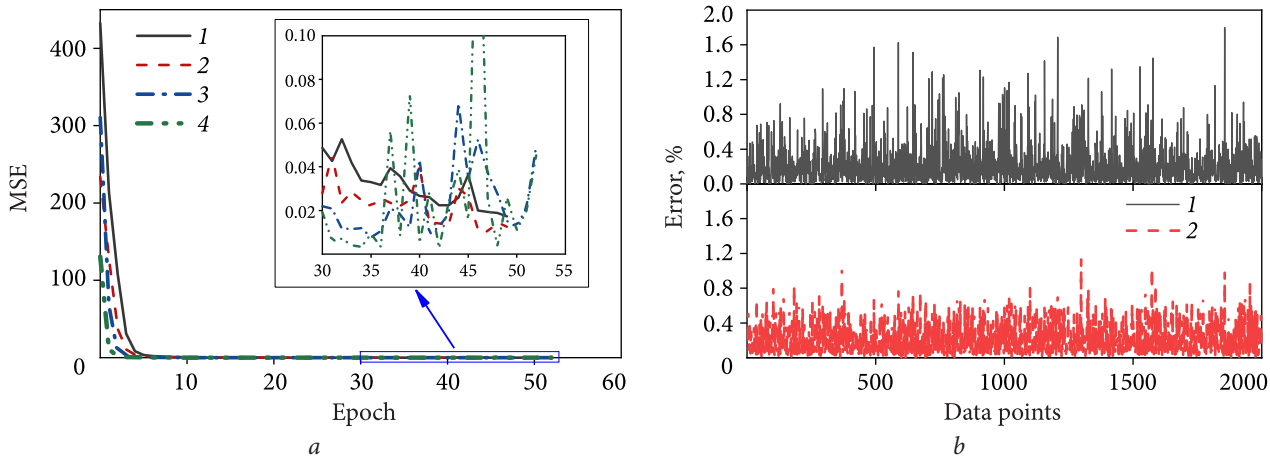


Fig. 7. Convergence rate (*a*) and relative error (*b*) for the frequencies predicted by the ANN-trained models differing in epoch counts and batch sizes: curves *1* and *2* in (*a*) and *1* in (*b*) — the epoch count reduction from 100 to 50, respectively; curves *3* and *4* in (*a*) and curve *2* in (*b*) — the batch size reduction from 128 to 64, respectively

causing divergence, this sensitivity level yields suboptimal model generalization.

The simplest of the three tested optimizers, Stochastic Gradient Descent (SGD), fails to train the ANN in the same terms (see curves *3* and *4* in Fig. 6, *a* and curve *2*, Fig. 6, *b*). Early in the training process, it generates NaN values, indicating instability in the gradient descent trajectory. This instability is likely due to the lack of adaptive learning rate adjustment, which is crucial for navigating over complex loss surfaces in regression problems, such as the resonant frequency prediction.

The results support a broad consensus among the machine-learning community that optimizers with adaptive learning mechanisms, such as the Adam, are better suited for tasks involving nonlinear correspondences [13, 27] and sensitive convergence dynamics. For high-dimensional input spaces and deep architectures, they offer improved stability and better performance with minimal tuning overhead.

2.4. Effect of Training Time and Batch Size

Two important hyperparameters that influence both the convergence dynamics and the final accuracy of the artificial neural network (ANN) are the batch size and the number of training epochs. In our computational experiments, cut each of them independently in half and assess the respective impacts.

Reduce the number of training epochs from 100 to 50. The model still converges but with diminished precision (curves *1* and *2* in Fig. 7, *a* and curve *1* in Fig. 7, *b*). The convergence rate remains high (the training convergence factor is 8.01), yet both training and validation errors increase slightly to 0.017 and 0.012, respectively. The reduction in training time is insignificant, suggesting that for this specific task, longer training yields more accurate predictions at a relatively low computational cost. The hypothesis is that additional epochs allow the model to examine weight parameter adjustments and better correlate the non-linear function driven by geometry parameters with resonant frequency.

In contrast, halving the batch size from 128 to 64 has a more noticeable effect (see curves *3* and *4* in Fig. 7, *a* and curve *2* in Fig. 7, *b*). Although the convergence rate remains nearly the same as before, the training error increases and, at best, reaches 0.0231, while the validation error remains relatively low (0.0117). The choice lies in balancing smaller data batches against noisier gradients to prevent slowing down convergence or model overfitting. In certain contexts, smaller batches can help prevent getting stuck in local minima because of the inherent stochasticity of the process. However, in our situation, the benefits of smaller batches do not outweigh the drawbacks. Therefore, a batch size of 128 seems optimal as it balances gradient stability with training efficiency.

These observations reinforce the importance of tailoring training dynamics depending on the problem complexity and available data. While reducing the computational load by decreasing the number of epochs or the batch size may seem advantageous, it can undermine model performance unless carefully balanced.

Conclusions

The study has demonstrated that both ANN architecture and training configuration critically determine the ANN performance in predicting resonant frequencies of rectangular patch antennas.

It has been revealed that the ANN depth plays a vital role. A shallow network with too few hidden layers lacks the representational capacity to capture the underlying physical dependences, resulting in higher prediction errors. In contrast, excessively deep architectures improve predictive accuracy to some extent but increase computational costs and the risk of overfitting. An optimal architecture in the considered case consists of three layers with [1024, 512, 256] neurons, which provides a good balance between performance and training time.

Activation functions substantially impact both convergence behavior and final model accuracy. ReLU emerged as the most stable and effective choice. The sigmoid activation pushed the model towards divergence and training collapse. Tahn is functional but led to less stable learning and higher errors. The findings emphasize the importance of the activation function selection for regression-oriented tasks.

The choice of the optimizer proved crucial. The adaptive Adam optimizer consistently delivered the best results in both accuracy and convergence rate. RMSprop was performing moderately well but less efficiently. SGD wholly failed model training. The importance of using adaptive learning rate techniques for complex nonlinear problems, specifically for resonant frequency prediction, is evident.

Lastly, training configuration parameters such as batch size and number of epochs also influence the final results. Reducing the number of epochs prevents the ANN's complete convergence. A smaller batch size increases loss function instability. To smooth training dynamics and improve prediction accuracy, a sufficiently high number of training steps and a moderate batch size are essential.

Overall, the findings emphasize that efficient electromagnetic modeling requires jointly optimizing the ANN architecture and training strategy and propose practical guidelines for AI-driven antenna designing.

Acknowledgments. *The authors are grateful to the National Research Foundation of Ukraine for the support under project grant no. 2025.06/0108.*

REFERENCES

1. James, J.R., Hall, P.S., Wood, C., 1981. *Microstrip Antenna. Theory and Design*, London: Peter Peregrinus LTD.
2. Kumar, G., Kamala, P.R., 2003. *Broadband Microstrip Antennas*. Boston, London: Artech House.
3. Waterhouse, R.B., 2013, *Microstrip Patch Antennas: A Designer's Guide*. New York: Springer Science & Business Media.

4. Patnaik, A., Anagnostou, D.E., Mishra, R.K., Christodoulou, C.G., Lyke, J.C., 2004. Applications of neural networks in wireless communications, *IEEE Antennas and Propagation Magazine*, **46**(3), pp. 130—137. DOI: 10.1109/MAP.2004.1374125
5. Feng, F., Zhang, J., Na, W., Zhang, W., Jin, J., Zhang, Q.-J., 2022. Artificial neural networks for microwave computer-aided design: The state of the art. *IEEE Trans. Microw. Theory Tech*., **70**(11), pp. 4597—4619. DOI: 10.1109/TMTT.2022.3197751
6. Yu, Y., Zhang, Z., Cheng, Q.S., Liu, B., Wang, Y., Guo, C., Ye, T.T., 2022. State-of-the-art: AI-assisted surrogate modeling and optimization for microwave filters. *IEEE Trans. Microw. Theory Tech*., **70**(11), pp. 4635—4651. DOI:10.1109/TMTT.2022.3208898
7. Qi, S., Sarris, C.D., 2022. Deep neural networks for rapid simulation of planar microwave circuits based on their layouts. *IEEE Trans. Microw. Theory Tech*., **70**(11), pp. 4805—4815. DOI: 10.1109/TMTT.2022.3210229
8. Swaminathan, M., Bhatti, O.W., Guo, Y., Huang, E., Akinwande, O., 2022. Bayesian Learning for Uncertainty Quantification, Optimization, and Inverse Design. *IEEE Trans. Microw. Theory Tech*., **70**(11), pp. 4620—4634. DOI: 10.1109/TMTT.2022.3206455
9. Ma, J., Dang, S., Li, P., Watkins, G., Morris, K.A., Beach, M.A., 2023. A learning-based methodology for microwave passive component design. *IEEE Trans. Microw. Theory Tech*., **71**(7), pp. 3037—3050. DOI: 10.1109/TMTT.2023.3238418
10. Thakare, V.V., Singhal, P., 2010. Microstrip antenna design using artificial neural networks. *Int. J. RF Microw. Comput.-Aided Eng.*, **20**(1), pp. 76—86. DOI: 10.1002/mmce.20414
11. Rawat, A., Yadav, R.N., Shrivastava, S.C., 2012. Neural network applications in smart antenna arrays: A review. *AEU — Int. J. Electron. Commun.*, **66(**11), pp. 903—912. DOI: 10.1016/j.aeue.2012.03.012
12. Khan, T., De, A., 2015. Modeling of microstrip antennas using neural networks techniques: a review. *Int. J. RF Microw. Comput.-Aided Eng.,* **25**(9), pp. 747—757. DOI: 10.1002/mmce.20910
13. Shi, D., Lian, W., Cui, K., Leong, D.S.H., 2022. An intelligent antenna synthesis method based on machine learning. *IEEE Trans. Antennas Propag.*, **70**(7), pp. 4965—4976. DOI: 10.1109/TAP.2022.3182693
14. Massa, A., Salucci, M., 2022. On the design of complex EM devices and systems through the system-by-design paradigm: A framework for dealing with the computational complexity. *IEEE Trans. Antennas Propag.*, **70**(2), pp. 1328—1343. DOI: 10.1109/TAP.2021.3111417
15. Raya, M.B., Pal, S., Ali, K., 2019. Design of inset fed rectangular shaped microstrip patch antenna using deep neural networks. In: *Proc. 2019 22nd Int. Conf. on Computer and Information Technology* (ICCIT). Dhaka, Bangladesh, 18—20 Dec. 2019. Dhaka: IEEE, 2019. DOI: 10.1109/ICCIT48885.2019.9038284
16. Pal, S., Raya, M.B., Ali, K., 2019. Computation of resonant frequency and gain from inset fed rectangular shaped microstrip patch antenna using deep neural network. In: *Proc. 2019 4th Int. Con. on Electrical Information and Communication Technology* (EICT). Khulna, Bangladesh, 20—22 Dec. 2019. Khulna. DOI: 10.1109/EICT48899.2019.9068758
17. Syahrial, S., Al Faqi, M.K., Meutia, E.D., Munadi, R., Roslidar R., 2024. Parameter estimation of two-element rectangular microstrip patch antenna using artificial neural network. In: *Proc. 2024 FORTEI-Int. Conf. on Electrical Engineering* (FORTEI-ICEE). Badung, Indonesia, 24—25 Oct. 2024. Badung: IEEE, pp. 216—221. DOI: 10.1109/FORTEI-ICEE64706.2024.10824537
18. Shereen, M.K., Liu, X., Wu, X., Niazi, S., Naseem, A., Khattak, M.I., 2025. Towards bi-directional deep learning approach in patch antenna design and optimization. In: *Proc. 2025 IEEE Int. Workshop on Antenna Technology* (iWAT). Cocoa Beach, FL, USA. 19—21 Feb. 2025. Cocoa Beach: IEEE. DOI: 10.1109/iWAT64079.2025.10931210
19. Derneryd, A., Lind, A., 1979. Extended analysis of rectangular microstrip resonator antennas. *IEEE Trans. Antennas Propag.*, **27**(6), pp. 846—849. DOI: 10.1109/TAP.1979.1142206
20. Garg, R., Bhartia, P., Bahl, I.J., Ittipiboon, A., 2001. *Microstrip Antenna Design Handbook*. Boston, London: Artech House.
21. Steer, M., 2019. *Fundamentals of Microwave and RF Design*, 3rd ed. USA: NC State University.
22. Fesenko, V.I., Tkachenko, G.V., 2007. Modeling of 1-D photonic bandgap microstrip structures. In: *Proc. 2007 Int. Workshop on Optoelectronic Physics and Technology* (OPT'07), Kharkiv, Ukraine, 20—22 June 2007, pp. 42—43. DOI: 10.1109/OPT.2007.4298524.
23. Mishra, R.K., Patnaik, A., 2003. Designing rectangular patch antenna using the neurospectral method. *IEEE Trans. Antennas Propag.*, **51**(8), pp. 1914—1921. DOI: 10.1109/TAP.2003.814748
24. Haykin, S., 2009. *Neural Networks and Learning Machines*. 3rd ed. NJ: Pearson Education.
25. Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep Learning*. Cambridge, Massachusetts London: MIT Press.
26. He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: *Proc. 2015 IEEE Int. Conf. on Computer Vision* (ICCV). Santiago, Chile, 7—13 Dec. 2015, pp. 1026—1034. DOI: 10.1109/ICCV.2015.123
27. Maity, B., Nayak, S.K., 2024. Artificial neural networks for optimal design parameters prediction of microstrip patch antenna with wideband harmonic suppression. In: *Proc. 2024 Second Int. Conf. on Microwave, Antenna and Communication* (MAC). Dehradun, India, 04—06 Oct. 2024. DOI: 10.1109/MAC61551.2024.10837513

Received 17.10.2025

С. Бутов[1], *А. Туз*[2], *О. Морозова*[2], *С. Вінніченко*[1], *О. Хардіков*[1], *С. Шульга*[1]
[1] Факультет радіофізики, біомедичної електроніки та комп'ютерних систем,
Харківський національний університет імені В.Н. Каразіна
майдан Свободи, 4, м. Харків, 61022, Україна
[2] Кафедра кібербезпеки та інтелектуальних інформаційних технологій,
Національний аерокосмічний університет «Харківський авіаційний інститут»
вул. Вадима Манька, 17, м. Харків, 61070, Україна

ОБЧИСЛЮВАЛЬНА ЕФЕКТИВНІСТЬ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ОПТИМІЗАЦІЇ УМОВ РОБОТИ МІКРОСМУЖКОВИХ АНТЕН

Предмет і мета роботи. Проєктування антен із заданими експлуатаційними характеристиками є складною задачею оптимізації з великою кількістю параметрів і нелінійних залежностей. Штучні нейронні мережі (ШНМ) продемонстрували значний потенціал в антенній інженерії завдяки здатності апроксимувати нелінійні функції та виявляти приховані взаємозв'язки, що забезпечує часткову автоматизацію процесу проєктування антен. Водночас це потребує детального аналізу впливу параметрів ШНМ на точність прогнозування та обчислювальну ефективність. Метою дослідження є вивчення впливу архітектури ШНМ і конфігурації навчання на точність передбачення резонансної частоти прямокутних мікросмужкових патч-антен.

Методи та методологія. Для оцінювання ефективності ШНМ за різних конфігурацій було розроблено модульну експериментальну платформу на базі Python. Навчання мереж здійснювалося на синтетичному наборі даних, згенерованому на основі класичної аналітичної моделі антени. Кількість прихованих шарів і нейронів у кожному шарі, типи функцій активації та оптимізатори систематично варіювалися з метою оцінювання їхнього впливу на збіжність, здатність до узагальнення та час виконання.

Результати. Отримані результати свідчать, що тришарова нейронна мережа з конфігурацією [1024, 512, 256] нейронів, функцією активації ReLU та оптимізатором Adam забезпечує найкращий баланс між точністю прогнозування та швидкістю навчання. Простіші або надмірно глибокі архітектури, неадаптивні оптимізатори та насичувальні функції активації призводять до уповільненої збіжності або нестабільного навчання. Додатковий аналіз показав, що зменшення розміру пакету підвищує стохастичність процесу навчання, проте може знижувати його стабільність.

Висновки. Дослідження демонструє, що спільна оптимізація архітектури ШНМ і динаміки процесу навчання є необхідною умовою створення точних і обчислювально ефективних електромагнітних моделей. Наведено практичні рекомендації щодо застосування методів машинного навчання в антенному проєктуванні.

Ключові слова: *штучна нейрона мережа, функція активації, адаптивний оптимізатор, мікросмужкова патч-антена, автоматичне проєктування антен.*