World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Research Article)

Deriving mathematical models from neural networks: A method for deducing individual effects of factors on a response variable

Henry Samambgwa [*], Thomas Musora and Joseph Kamusha

Department of Mathematics and Statistics, Chinhoyi University of Technology, Chinhoyi, Zimbabwe. Private Bag 7724, Chinhoyi, Zimbabwe.

World Journal of Advanced Engineering Technology and Sciences, 2025, 17(03), 051-059

Publication history: Received 20 October 2025; revised on 29 November 2025; accepted on 01 December 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.17.3.1524

Abstract

This research outlines a novel approach to obtaining mathematical models from neural networks. The target scenario is one where a response variable depends on a number of factors, each factor has an effect which is a function of the factor and the response variable is the sum of the effects of the factors. A neural network was trained such that response values were generated from factor values. It was assumed that each effect was zero when the underlying factor was set to zero. The effect of a factor could be isolated by setting all other factors to zero, so that the response value became equal to the effect of the factor being isolated. In that way each effect was isolated and then modelled as a function of the factor. Thus, the technique was developed, for modelling a response variable as a function of its input factors.

Keywords: Neural Network; Perceptron; Mathematical Modelling; Machine Learning; Simulation

1. Introduction

Neural network research is a subfield of machine learning, concerned with replication of human brain function in machines [1]. Machine learning procedures are designed to learn from empirical data [2]. Machine learning itself is a subfield of artificial intelligence (AI). AI research is concerned with the replication of human actions, activities and abilities in machines. Machines are designed to replicate human hearing, sight, touch, thinking and human activity recognition (HAR). In HAR, machines are taught to monitor and analyse human movements using sensor or visual data [3]. AI has transformed various sectors by integrating human-like abilities such as learning, reasoning, and perception into software systems [4].

Artificial neural network research was developed in the 1940s, by researchers Warren McCulloh and Walter Pitts, who proposed a mathematical model for brain neurons [5]. Over the decades since, neural network research has developed tremendously and made valuable contributions to academia, mining, pharmaceutical, medical, financial and geographical research among many other fields of study [6].

1.1. Neural networks

Neural networks are mathematical models were input variable data is propagated from the input layer neurons, through hidden layer neurons and then response data is obtained at the output layer neurons. Neural networks are inspired by the human brain, and they replicate the way real neurons communicate with one another [7]. Neural networks have been studied under statistical mechanics since the 1980s [8], and have proven to be a highly effective tool for solving complex problems in many areas of life [9]. A neuron calculates the weighted sum of its inputs and then applies an activation function to obtain a signal that will be transmitted to the next neuron [10]. Neural networks reduce

[*] Corresponding author: Henry Samambgwa

humanity's burden to solve complex problems highly efficiently [11]. Figure 1 illustrates the basic structure of a feedforward neural network.

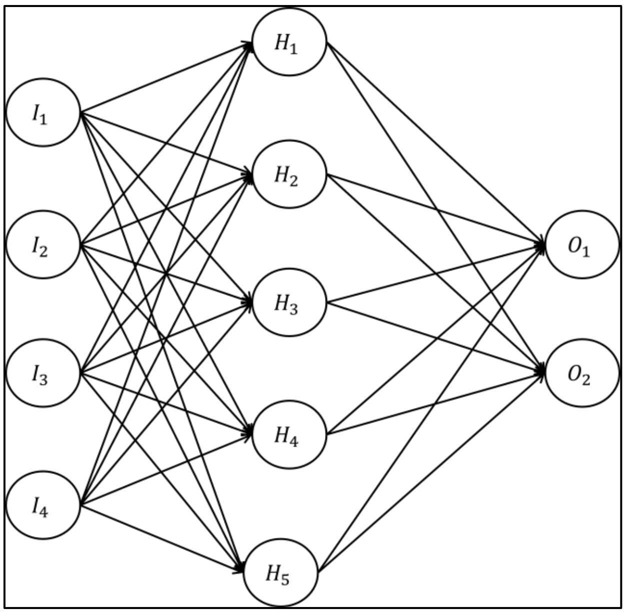


Figure 1 Feedforward neural network

In this neural network the input layer (left) has four neurons and the output layer(right) has two neurons. Values from input variables are set in the input layer. Values of the response variables are generated at the output layer. In general, neuron layers between the input and the output layers are called hidden layers [12]. Here there is one hidden layer with five neurons.

Input values are captured at the neurons in the input layer. Each neuron in the input layer passes its value to each neuron in the hidden layer. All neurons in hidden layers collect values from each neuron in the preceding layer and pass values to each neuron in the succeeding layer. The values coming into each hidden layer neuron are weighted and summed.

1.2. Input

Equation (1) shows how the weighted sum of inputs y_{H_i} is calculated at each neuron H_i.

$$y_{H_i} = \sum_{j=1}^{4} w_{I_j H_i} x_{I_j}\,, 1 \leq i \leq 5, \qquad (1)$$

were

- y_{H_i}= weighted sum of inputs to neuron H_i,
- x_{I_j}= input from neuron I_j,
- $w_{I_j H_i}$= weight of the input from neuron I_j to neuron H_i.
- Each arrow connecting neurons has a weight associated with it.

The weighted sum is then transformed (activated) using an activation function. This is done in order to remove linearity. If activation was not done, the final output would be a linear combination of the initial inputs. This would restrict the neural network to modelling linear relationships. Activation removes linearity, allowing the neural network to model a wider range of relationships including complicated functions. This widens the applicability of neural networks.

1.3. Activation

There are numerous functions that are used for activating neurons. Common activation functions include the sigmoid, tanh, RELU and SOFTMAX among many others [13]. The choice of activation function can impact the accuracy of the resulting neural network model [14]. The sigmoid function can be used to estimate any continuous function using the neural network [15]. Activation transforms the weighted sum y_{H_i} into values which are then passed into the next layer. Equation (2) shows how the result, x_{H_i}, at a neuron, H_i is calculated using the activation function, A.

$$x_{H_i} = A(y_{H_i}), \qquad (2)$$

were

- x_{H_i}=input from neuron H_i to the next layer,
- A= activation function,
- y_{H_i}= weighted sum of inputs to neuron H_i.
- Activation occurs at each neuron in the hidden layer and output layer.

1.4. Multilayer Perceptron

The multilayer perceptron (MLP) is the most known and most frequently used type of neural network [16]. It is the most fundamental and important neural network model [17]. A perceptron has one neuron in the output layer. The MLP is very efficient for function approximation in high-dimensional spaces [18]. Perceptron neural networks produce efficient solutions to problems of overwhelming complexity [19].

1.5. Simulating clandestine functions

Neural networks are trained using empirical data. Training algorithms adjust the weights accordingly so that the resulting neural network replicates the causality of input data to response data [20]. Neural networks are used to model unknown causality relationships where empirical values of input and response variables are known.

1.6. Capturing Multiple influences

This research focused on situations where a single response variable is known to depend on several factors. Neural networks can model systems where the particular effects (influences or contributions) of each factor are not known. In this study, the researchers developed a method to isolate and model those individual effects using a trained neural network.

2. Methodology

This study develops a method to derive a mathematical model from a neural network. We assume that the response variable (R), which is a function of n factors, is the sum of the individual effects (E_i) of the factors, where each effect is a function of the underlying factor. Equation (3) illustrates this

$$R(F_1, F_2, F_3, \ldots, F_n) = E_1(F_1) + E_2(F_2) + E_3(F_3) + \ldots + E_n(F_n), \qquad (3)$$

were

- R= the response variable which is a function of the factors $F_1, F_2, F_3, \ldots, F_n$,
- $E_i(F_i)$=the effect of factor F_i.

We further assume that the factors are independent and the effects are independent of each other. We also assume that each factor has no effect when it is zero (Equation (4)).

- $E_i(0) = 0,\ \forall i \in \{1, 2, 3, \ldots, n\}$. (4)

A neural network is trained using empirical data. The neural network then replicates the effect of the factors on the response variable. To deduce the effect of an individual factor (F_m), we set all other factors (F_i) to zero, (where $i \neq m$). We then run the neural network, with only F_m active. The result, thus only depends on F_m and is equal to the individual effect of F_m,(Equations (5) and (6)).

- $R(F_1, F_2, F_3, \ldots, F_n) = E_1(F_1) + E_2(F_2) + E_3(F_3) + \ldots + E_n(F_n)$ (5)
- $R(0,0,0, \ldots, F_m, \ldots, 0) = E_1(0) + E_2(0) + E_3(0) + \ldots + E_m(F_m) \ldots + E_n(0) = E_m(F_m)$ (6)

Having isolated the effect of F_m we then fit an appropriate function with F_m as the input variable and R as the response. In this way we model the individual effect of the factor. The method can then be repeated on other factors individually, allowing us to model all the effects of the factors. When added, the functions of the effects form a mathematical model of the response variable as a function of the factors (Equation (7)):

- $R(F_1, F_2, F_3, \ldots, F_n) = E_1(F_1) + E_2(F_2) + E_3(F_3) + \ldots + E_n(F_n)$ (7)

3. Analysis and Results

3.1. Introduction

An experiment was conducted to test the method. The experiment was conducted in the MATLAB online environment [21]. Three functions were used as effects:

- A polynomial function(cubic),
- a trigonometric function,
- and an exponential function.

The model used for testing is shown in Equation (8)

$$R(F_1, F_2, F_3) = 5F_1^3 - 11F_1^2 + 5F_1 + 20sinF_2 + 3F_3e^{F_3} \qquad (8)$$

$Figure$ 2 shows the MATLAB script that was used to generate random data and train a neural network with the generated data.

10 000 Random values of F_1, F_2 and F_3 were generated within the following domains:

$$-5 \leq F_1 \leq 5$$

$$0 \leq F_2 \leq 10\pi$$

$$-10 \leq F_3 \leq 10$$

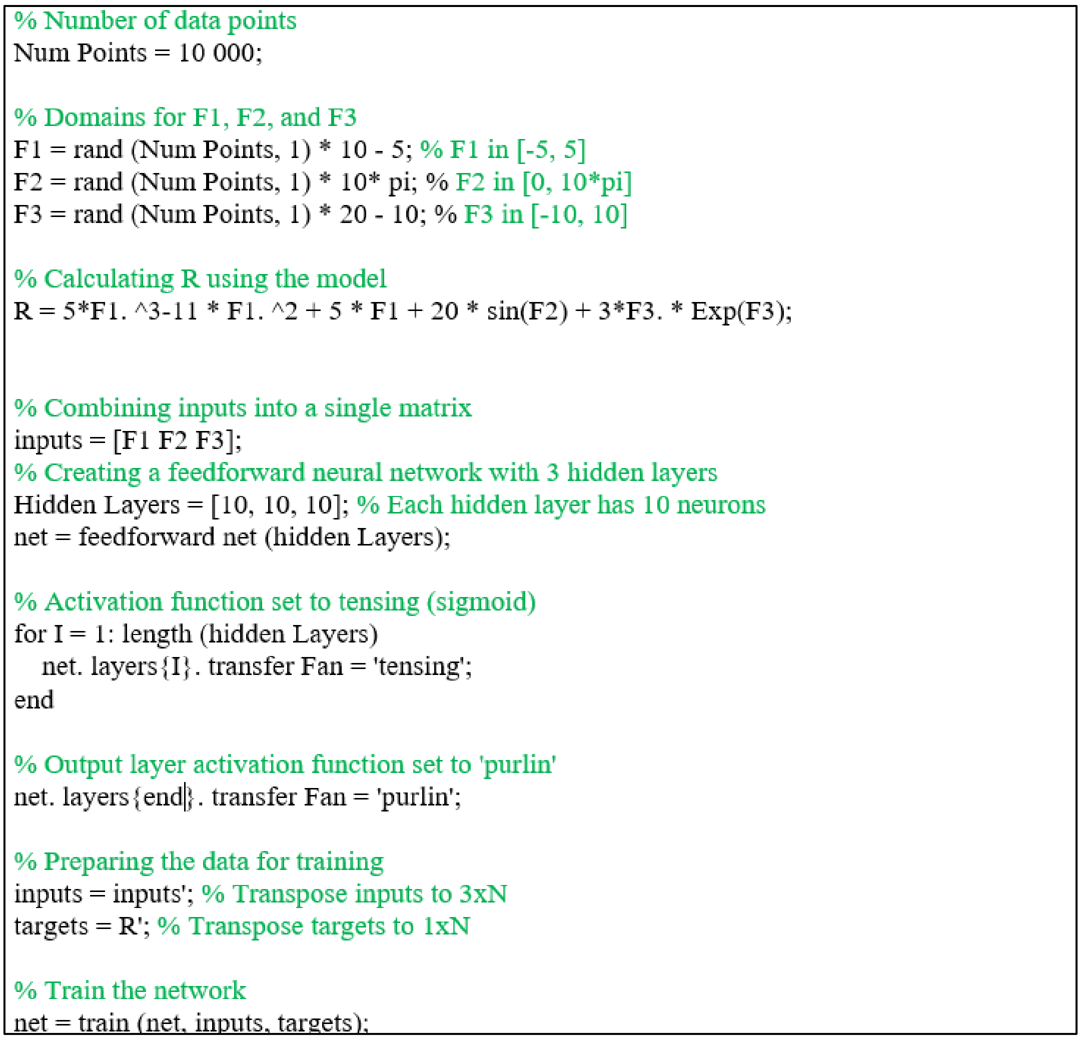
Corresponding values of R were calculated using the model formula.

The neural network was setup with 3 hidden layers of 10 neurons each and trained using the generated data. The input layer had three inputs (F_1, F_2 and F_3) and the output layer had one output (R). The $tansig$ (sigmoid) activation function (Equation (9)) was used in hidden layers:

$$tansig(x) = \frac{1}{1+e^{-x}}. \qquad (9)$$

The output layer was setup with the $purelin$ activation function (Equation (10))

$$purelin(x) = x. \qquad (10)$$



```
% Number of data points
Num Points = 10 000;

% Domains for F1, F2, and F3
F1 = rand (Num Points, 1) * 10 - 5; % F1 in [-5, 5]
F2 = rand (Num Points, 1) * 10* pi; % F2 in [0, 10*pi]
F3 = rand (Num Points, 1) * 20 - 10; % F3 in [-10, 10]

% Calculating R using the model
R = 5*F1. ^3-11 * F1. ^2 + 5 * F1 + 20 * sin(F2) + 3*F3. * Exp(F3);


% Combining inputs into a single matrix
inputs = [F1 F2 F3];
% Creating a feedforward neural network with 3 hidden layers
Hidden Layers = [10, 10, 10]; % Each hidden layer has 10 neurons
net = feedforward net (hidden Layers);

% Activation function set to tensing (sigmoid)
for I = 1: length (hidden Layers)
   net. layers{I}. transfer Fan = 'tensing';
end

% Output layer activation function set to 'purlin'
net. layers{end}. transfer Fan = 'purlin';

% Preparing the data for training
inputs = inputs'; % Transpose inputs to 3xN
targets = R'; % Transpose targets to 1xN

% Train the network
net = train (net, inputs, targets);
```

Figure 2 MATLAB script used for data generation and neural network training

3.2. Effect of F_1

The effect of F_1 was determined using the MATLAB script in Table 1. The script functioned as follows:

The variables F_2 and F_3 were set to zero whilst 10 000 values were generated for F_1 using the function in Equation (11)

$$F_1 = -5 + \frac{i}{1000},\ i \in \{x: 1 \leq x \leq 10\,000, x \in N\}. \qquad (11)$$

The neural network was simulated using the matrix of generated values (Equation (12)):

$$input = \begin{bmatrix} -4.999 & 0 & 0 \\ -4.998 & 0 & 0 \\ -4.997 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 5.000 & 0 & 0 \end{bmatrix}. \qquad (12)$$

The simulation generated corresponding values of R. The MATLAB $polyfit$ function was then used to fit a polynomial of degree 5 with F_1 as the sole input variable and R as the response variable:

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f \qquad (13)$$

The resulting model is shown in Equation (14):

$$R(F_1, 0{,}0) = 0.0004F_1{}^5 \ + \ 0.0005F_1{}^4 + 4.9905F_1{}^3 \ - 11.0073F_1{}^2 + \ 5.0391F_1 \ - 0.0078 \qquad (14)$$

Which was approximately equal to the actual effect of F_1. The method thus succeeded in determining the effect of F_1.

Table 1 MATLAB script used to generate values of $\boldsymbol{F_1}$, simulate the neural network and fit a polynomial of degree 5 to the resulting data

MATLAB script	Output
%Generating values of F_1 in [-5,5] F_1=-5+(10/10000) *[1:10000]; %Setting up the input matrix input= [F_1' zeros (10000,1) zeros (10000,1)]; %Simulating the neural network to generate values of R_1 R_1=net(in'); %Fitting a polynomial of degree 5 to F_1 against R_1 polykite (F_1, R_1,5)	Ans = 0.0004 0.0005 4.9905 -11.0073 5.0391 -0.0078

3.3. Effect of $\boldsymbol{F_2}$

The effect of $F_2$was determined using the MATLAB script in Table 2. The script functioned as follows:

The variables F_1 and F_3 were set to zero whilst 10 000 values are generated for F_2 using the function in Equation (15):

$$F_2 = \frac{10\pi i}{10\,000}, \ i \in \{x: 1 \leq x \leq 10\,000, x \in N\} \qquad (15)$$

The neural network was simulated using the matrix of generated values (Equation (16))

$$input = \begin{bmatrix} 0 & \frac{1}{1000}\pi & 0 \\ 0 & \frac{2}{1000}\pi & 0 \\ 0 & \frac{3}{1000}\pi & 0 \\ \vdots & \vdots & \vdots \\ . & . & . \\ 0 & 10\pi & 0 \end{bmatrix} \qquad (16)$$

The simulation generated corresponding values of R. The MATLAB $fittype$ function was used with the general form $asin(bx \ + \ c) + d$ specified for the function.

The resulting function is shown in Equation (17)

$$R(0, F_2, 0) = 19.99sin(F_2 - \ 0.00005935) - 0.02509 \qquad (17)$$

This is approximately equal to the actual effect of F_2. The method thus succeeded in determining the effect of F_2.

Table 2 MATLAB script used to generate values of $\boldsymbol{F_2}$, simulate the neural network and fit a trigonometric model to the resulting data

MATLAB script	**Output**

%Generating 10000 values of F_2 in [0,10*pi]
F_2 = (10*pi/10000) *[1:10000]';
%Setting the input matrix with F_2 active and F_1 and F_3 zero
input= [zeros (10000,1) F_2 zeros (10000,1)];
	General model:
	fit Result(x) = a*sin(b*x + c) + d
	Coefficients (with 95% confidence bounds):
	a = 19.99 (19.99, 20)

%Simulating the neural network
R_2=net(input')';
% Fit type for a trigonometric function
ft = fittype ('a*sin (b*x + c) + d', 'independent','x','dependent','y');
 	
% Fitting the model to the data
initialGuess = [1, 1, 0, 0];
[fitResult, gof] = fit (F_2, R_2, ft, 'StartPoint', initialGuess);
 	
% Display of the fit results
disp(fitResult);
disp(gof);
	b = 1 (1, 1)
	c = -5.935e-05 (-0.0004651, 0.0003464)
	d = -0.02509 (-0.02796, -0.02221)
	sse: 214.3639
	rsquare: 0.9999
	dfe: 9996
	adjrsquare: 0.9999
	rmse: 0.1464

3.4. Effect of F_3

The effect of $F_3$was determined using the MATLAB script in Table 3 . The script functioned as follows:

The variables F_1 and F_2 were set to zero whilst 10 000 values were generated for F_3 using the function in Equation (18):

$$F_3 = -10 + \frac{20i}{10\,000},\ i \in \{x: 1 \le x \le 10\,000, x \in N\} \qquad (18)$$

The neural network was simulated using the matrix of generated values (Equation (19))

$$input = \begin{bmatrix} 0 & 0 & -9.998 \\ 0 & 0 & -9.996 \\ 0 & 0 & -9.994 \\ \vdots & \vdots & \vdots \\ . & . & . \\ 0 & 0 & 10.000 \end{bmatrix}. \qquad (19)$$

The simulation generated corresponding values of R. The MATLAB $fittype$ function was used with the general form ae^{bx} specified for the function.

The resulting function is shown in Equation (20):

$$R(0,0,F_3) = 3e^{bF_3}. \qquad (20)$$

Which is precisely equal to the actual effect of F_3.

Table 3 MATLAB script used to generate values of $\boldsymbol{F_3}$, simulate the neural network and fit an exponential model to the resulting data

MATLAB script	**Output**

%Generating 10000 values of F_3 in [-10,10]
F_3 = -10+(20/10000) *[1:10000]';
%Setting the input matrix with F_3 active and F_1 and F_2 zero
input= [zeros (10000,1) zeros (10000,1) F_3];
%Simulating the neural network
R_3=net(input')';
% Fit type for an exponential function
ft = fittype('a*x*exp(b*x)', 'independent','x','dependent','y');
	General model:
	fitResult(x) = a*x*exp(b*x)
	Coefficients (with 95% confidence bounds):
	a = 3 (3, 3)
	b = 1 (1, 1)
	sse: 6.8685e+04
	rsquare: 1.0000
	dfe: 9998

% Fitting the model to the data
initialGuess = [0,0];
[fitResult, gof] = fit (F_3, R_3, ft, 'StartPoint', initialGuess);
 	
% Display of the fit results
disp(fitResult);
disp(gof);
	adjrsquare: 1.0000
	rmse: 2.6210

3.5. Identifying the General Form

When using this technique, it is necessary to plot value of the factor against it's the generated response (equal to the effect) values in order to identify the type of function and thus determine a suitable general form.

4. Discussion

In life there are systems in which a response variable results from the sum of effects of various factors. The aim of this study was to develop a technique that can isolate and model each individual effect in such systems as a function of the factor causing it. The researchers simulated one such scenario and the experimental results demonstrated the functionality of the technique. The technique can be used solve problems with similar objectives.

This research demonstrated that neural network models replicate the true individual components of the system that they are trained for. An alternative hypothesis would have been that neural networks develop operations that replicate the attainment of similar responses from similar inputs as the systems they represent, without necessarily forming the true underlying causalities. In this study it was thus demonstrated that neural networks learn the actual operations of the systems that they are trained for. This proposes that neural networks hold within them, the secret operations of the systems they represent. It is thus up to researchers to derive those secrets from the neural networks.

The technique was demonstrated for polynomial, trigonometric and exponential functions. Further research may be pursued to test the functionality for other types of functions. The technique can be used to deduce functions where it would be infeasible, hazardous or expensive to directly measure and analyse the effects in their real-life situations.

5. Conclusion

This study successfully developed a novel technique for isolating and modelling the individual effects of various factors in complex systems. It was demonstrated that neural networks replicate the actual underlying processes of the systems they are trained on. When analytic models of underlying processes are deduced, further insights can be drawn on the form and manner of the influences determining the system. The ability to accurately deduce individual contributions from aggregate effects opens new avenues for research, particularly in scenarios where direct measurement is challenging or impractical. Future research should aim to test the technique on additional types of functions, potentially expanding its reach and enhancing our understanding of complex systems. Ultimately, the findings suggest a promising direction for the application of neural networks in data analysis, offering researchers a powerful tool to unlock the secrets of intricate causal relationships in various domains.

Compliance with ethical standards

Disclosure of conflict of interest

All authors declared they have no conflicts of interest related to this research.

References

[1] Goodfellow I., Bengio Y., and Courville A. (2016). Deep Learning. MIT Press.

[2] Musora, T. (2023). Application of Panel Data Analysis and Modelling to Economic Data: A Case of Determinants of Economic Growth for SADC and Zimbabwe. Chinhoyi University of Technology.

[3] Sennan, S., Somula, R., Cho, Y., Pandey, B. K. (2025). A hybrid multi-layer perceptron with selective stacked ensemble learning approach for recognizing human activity using sensor dataset. Scientific Reports.

[4] Przybyla-Kasperek, M., Marfo, K. F. (2024). A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data. PLoS ONE.

[5] McCulloch W., and Pitts W. (1943). A logical calculus of the ideas immanent in nervous activity.

[6] The Bulletin of Mathematical Biophysics.

[7] LeCun Y., Bengio Y., and Haffner P. (2015). Gradient-based learning applied to document recognition. Proceedings of the Institute of Electrical and Electronics Engineers.

[8] Qamar, R., Zardari, B. A. (2023). Artificial Neural Networks: An Overview. Mesopotamia Journal of Computer Science.

[9] Gabrie, M., Ganguli, S., Lucibello, C., Zecchina, R. (2023). Neural Networks: from the perceptron to deep nets. Replica Symmetry Breaking and Far Beyond.

[10] Kunc, V., Klema, J. (2024). Three decades of activation. A comprehensive survey of 400 activation functions for neural networks. Machine Learning. Cornell University.

[11] Castro, W., Oblitas, J., Santa-Cruz, R., Avilla-George, H. (2017). Multilayer perceptron architecture optimization using parallel computing techniques. PLoS ONE.

[12] Madhiarasan, M., Louzazni, M. (2022). Analysis of Artificial Neural Network: Architecture, Types, and Forecasting Applications. Journal of Electrical and Computer Engineering. Wiley.

[13] Apicella, A., Donnarumma, F., Isgro, F., Prevete, R. (2021). A survey on modern trainable activation functions. Neural Networks. Elsevier.

[14] Bishop C. (2006). Pattern Recognition and Machine Learning. Springer.

[15] Ding, B., Qian, H., Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. 2018 Chinese Control and Decision Conference.

[16] Kamalov, F., Nazir, A., Safaraliev, M., Cherukuri, A. K., Zgheib, R. (2021). Comparative analysis of activation functions in neural networks. International Conference on Electronics, Circuits and Systems. IEEE.

[17] Popescu, M. C., Balas, V. E., Popescu, L. P., Mastorakis, N. (2009). Multilayer Perceptron and Neural Networks. WSEAS Transactions on Circuits and Systems.

[18] Du, K. L., Leung, C. S., Mow, W. H., Swamy, M. N. S. (2022). Perceptron: Learning, Generalization, Model Selection, Fault Tolerance, and Role in the Deep Learning Era. Mathematics. MDPI.

[19] Du, K. L., Swamy, M. N. S. (2013). Neural Networks and Statistical Learning. Concordia University, Canada.

[20] Calude, C. S., Heidari, S., Sifakis, J. (2022). What perceptron neural networks are (not) good for? Information Sciences. Elsevier.

[21] Rumelhart D. E., Hinton G. E., and Williams R. J. (1986). Learning representations by back-propagating errors. Nature.

[22] MathWorks (2025). MATLAB. Accessed at: www.mathworks.com