www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(4), 2026, 269-286. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i4.1386

Design of a multilayer neural network in only two rows of an Excel spreadsheet for students without advanced programming knowledge

Roberto Baeza Serrato
Department of Multidisciplinary Studies, Engineering Division, Campus Irapuato-Salamanca, University of Guanajuato, Yuriria 38944, Guanajuato, Mexico.

Abstract. Artificial neural networks (ANNs) are intelligent techniques used as classification, prediction, or pattern recognition elements. It is necessary to have advanced programming knowledge to develop an artificial neural network using the Backpropagation (BP) learning algorithm. This research proposes the development of a multilayer neural network using the BP algorithm in only two rows of an Excel spreadsheet for students without advanced programming knowledge. The first row: the forward propagation of the network and the calculation of gradients in a simple manner. Each calculation is made in a specific cell. In the second row, the error is propagated backwards using the Backpropagation algorithm, adjusting the weights and biases of the output and hidden layers by subtracting the product of the two gradients. Once the correct filling of the two rows is complete, the data from the first and second rows is selected and dragged to the desired number of iterations in an easy, clear, and simple way, so that anyone without programming knowledge can develop a multilayer neural network. The tool does not require installation and offers a portable solution for designing neural networks for engineering students or students in any degree program without advanced programming knowledge. This methodology has been taught to 10 bachelor's and master´s groups in business management at the University of Guanajuato, resulting in 11 articles published in peer-reviewed journals, 3 bachelor's theses, and 4 master's theses.
Keywords: Artificial Neural network, Backpropagation algorithm, Excel spreadsheet, Gradients.

Article information
Received: April 11, 2026
Accepted: Jul 11, 2026

1 Introduction

Artificial Neural Networks represent an artificial intelligence technique with applications in pattern recognition, classification, and prediction. Its use is significant in various fields of engineering, such as electronic engineering, computing, chemistry, industrial, etc. Cao et al. (2018) mentioned that Artificial Neural Networks (ANNs) have received considerable attention due to their powerful ability in image processing, speech recognition, natural language processing, etc. The development of a neural network requires advanced programming knowledge. This represents an obstacle for undergraduate students who do not have that kind of knowledge, limiting practical access to this powerful technology in applied fields such as manufacturing, agriculture, or academic performance prediction (Lou et al., 2024; Chen et al., 2025; Liu et al., 2025). The standard method of implementing Artificial Neural Networks is by using C++, Python, or other high-level computer languages to develop a system able to take a set of inputs and generalize to produce a satisfactory output.

In this context, Excel emerges as an easy-to-use tool for any student of any discipline. Recent research highlights the potential of office tools and spreadsheets as accessible platforms for integrating artificial intelligence functionalities, facilitating learning and experimentation without a high technical barrier and classify volcanic rocks following IUGS recommendations entirely within the Microsoft Excel environment (White & Lucci, 2026). This research proposes the design and development of a neural network in an Excel spreadsheet, as an alternative tool for students who lack programming knowledge and wish to use this artificial intelligence technique to solve problems of classification, prediction, or pattern recognition in any professional field.

Excel is a tool that allows you to solve problems iteratively. The programming cycles carry out the necessary iterations until the objective is met. The forward propagation phases and gradients for the backpropagation algorithm are specified in the initial row of the Excel spreadsheet. The backward propagation of the error is performed from the second row, and the corresponding weights and biases of the neural network are adjusted, linking them to the gradients determined in the first row. The forward propagation phase and the backpropagation algorithm's gradients are then calculated. When this second line is finished, it is completely selected and dragged to the desired number of iterations specified by the network designer. The multi-layer neural network is ready to perform the training. This proposed approach, aligned with emerging frameworks that implement ANNs directly in spreadsheet environments, enables the development of a neural network for any student at any degree level, without the need for advanced programming knowledge in any language. The backpropagation algorithm is highly complex for students and professionals alike, and its subsequent application in programming. The proposed approach allows, visually and straightforwardly, for its operation to be explained by determining the gradients that adjust the weights and biases of the output and hidden layers of the multilayer neural network.

The use of accessible digital tools, such as spreadsheets, for teaching complex engineering and artificial intelligence concepts constitutes a growing area of pedagogical research. Traditionally, applications such as Excel have been demonstrated how common spreadsheet software can help with psychometric analysis. Authors used a sample dataset of 40 students, and 20 items were scored as right or wrong. Using simple spreadsheet formulas, we calculated basic statistics for both items and students, as well as indices from Classical Test Theory: item difficulty, discrimination, and corrected item–total correlations. (Faria & Mianda, 2026). Using spreadsheets as an introduction to programming helps students develop concept-based problem-solving skills. It also provides them with practical tools for real-world challenges, prepares them for database work, and supports their learning of advanced programming (Nagy et al., 2021). For four decades, spreadsheets have demonstrated substantial value in education, particularly in scientific and engineering fields. Building on this foundation, the present study introduces a didactic resource: the sprinkler irrigation tool, created with an Excel spreadsheet. This tool enables irrigation engineering students—especially those specializing in sprinkler irrigation system design to visualize layouts, calculate uniformity coefficients, and simulate scenarios, thereby enhancing theoretical understanding and practical skills. It is effective in both laboratory and field experiments, proving effective platforms for practical learning (Bautista-Capetillo et al., 2023).

This trend continues over time, as evidenced a linear regression algorithm based on Maximum Likelihood Estimation (MLE) was developed, accompanied by an Excel spreadsheet and a VBA program, specifically adapted for fracture-aperture data sets where sampling the smallest values poses challenges. Monte Carlo simulations demonstrate that this method achieves substantially better convergence compared to the Least Squares criterion (Guerriero, 2023). Similarly, in the field of chemical and biochemical engineering, native Excel functionalities, such as Solver, have been used to teach system identification using static and dynamic ANNs, offering students a practical introduction to AI models (Pinto et al., 2023). In the same way, Lee et al. (2025) proposed an Artificial Neural Network in Excel (ANN-Excel) software. This software implements a neural network-based prediction feature using Excel, making it easy to handle data in a spreadsheet and perform calculations with Visual Basic for Applications (VBA) code. The software has been implemented using a feed-forward multilayer perceptron, an error back-propagation algorithm, a momentum term, and data scaling, enabling users to use machine learning techniques directly in Excel without prior knowledge of the mechanisms.

The three previous quotes using solver or VBA programming work as functions. Users only need to enter the input and output data and define the number of neurons in the hidden layer. Students, however, do not understand the forward and backward propagation stages. In contrast, the present research allows students to acquire knowledge and develop any network structure or topology needed to solve the problem under study.

Concurrently, the integration of Machine Learning (ML) and ANNs in higher and professional education has been subject to significant scrutiny. A recent systematic review confirms that predicting academic performance is one of the most researched applications of ML in universities, highlighting the transformative potential of these technologies and the ethical challenges in the era of tools like ChatGPT (Pinto et al., 2023). This potential manifests in innovative approaches to teacher training, where deep learning architectures such as Convolutional Neural Networks (CNNs) and Transformers are used to create highly effective, personalized professional development programs (Li, 2025).

Visual communication design (VCD) is evolving, prompting changes in underlying design concepts. Traditional graphic design education is increasingly insufficient to address the demands of contemporary design education, necessitating reform to enhance the quality of VCD instruction. This study uses a graph neural network to examine the VCD course as a case study and identify effective strategies for VCD education reform (Wu et al., 2025). The relevance of these competencies extends to entrepreneurship,

where improved ANN algorithms have been developed to evaluate and enhance the quality of innovative education in universities (Sun & Zhang, 2022).

In advanced applications, ANNs have demonstrated superior performance in complex prediction and selection tasks, such as multilayer perceptron, convolutional neural networks, and recurrent neural networks, outperforming traditional methods (Guo et al., 2026). The article discusses the pros and cons of these networks, along with the most current f indings from their electronic nose applications. This underscores the technique's power, as previously recognized in earlier reviews of its capabilities for image and language processing (Cao et al., 2018).

However, a significant gap persists. Although proposals exist to implement ANNs in Excel, they still rely on add-ins such as VBA (Muneer & Ivanova, 2022) or the Solver tool (Ragsdale & Zobel, 2010), this study proposes a simple, effective method for extracting photovoltaic (PV) module parameters using Microsoft Excel's Solver. By employing the widely accessible Excel, users can straightforwardly implement the proposed method, which can pose a barrier for users without programming knowledge.

Therefore, a need is identified for a method that comprehensively implements the backpropagation algorithm for a multilayer neural network using only the native formulas and the standard Excel interface, without relying on macros, Solver, or any other external add-in. This approach would make network training tangible—from forward propagation to weight adjustment via gradients—by simply replicating rows, offering a universal, transparent, and very low-threshold pedagogical tool for students of any discipline. The present research directly addresses this need.

2. Methodology

The design and development of a multilayer neural network in an Excel sheet is presented. Two stages are developed for algorithm training: the forward-propagation stage and the learning stage with back-propagation of error. The forward propagation of the network takes place in 4 phases; 1) net input to the neurons of the hidden layer, 2) output of the neuron from the hidden layer using an activation function of the sigmoidal type, 3) net input to the neuron of the output layer and 4) output of the neuron of the output layer using an activation function of the sigmoidal type. These calculations are done in the second row of the Excel sheet. The gradients for the output and hidden layers are calculated in the same second row of the Excel sheet. Each tile is calculated in a specific cell.

The learning stage is developed in phases A and B, propagating the error backwards through the calculation of partial gradients. Phase A: Adjust the bias output layer with two gradients: gradient error output layer and gradient activation function output layer. Weight adjustment output layer with three gradients: gradient output layer error, gradient activation function output layer, and gradient weight output layer. Phase B: Adjust the bias hidden layer with two gradients: accumulated gradient error output layer and gradient activation function hidden layer. Finally, adjustments of hidden-layer weights with three gradients: accumulated gradient error output layer, gradient activation function hidden layer, and gradient weights hidden layer. The adjustments to the weights and biases of the output and hidden layers are made in the third row of the Excel sheet. Finally, the data is selected and dragged to the desired number of iterations. See Figure 1.

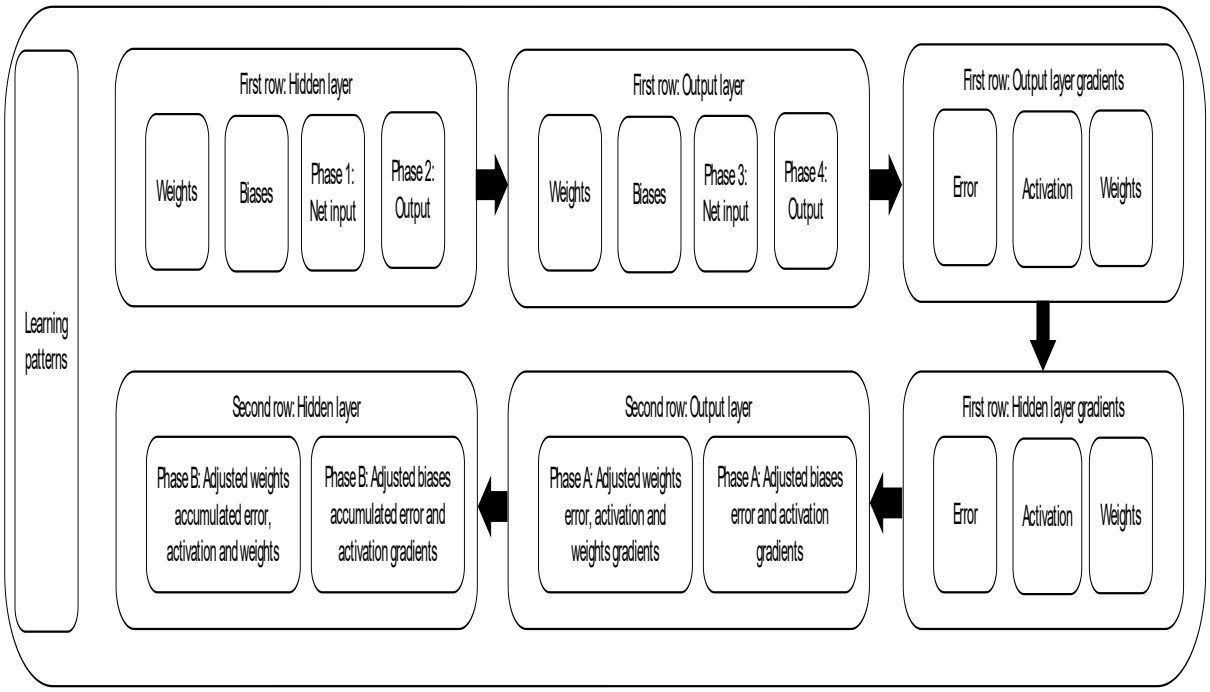


Figure 1. Neural multilayer network proposed

The forward propagation phase is shown in Figure 2, where two operations are performed on the hidden-layer neurons: first, a weighted sum, and then an activation function is applied to the weighted sum. Similarly, in the neuron of the output layer, both operations are performed: a weighted sum and the use of a sigmoid-type activation function.

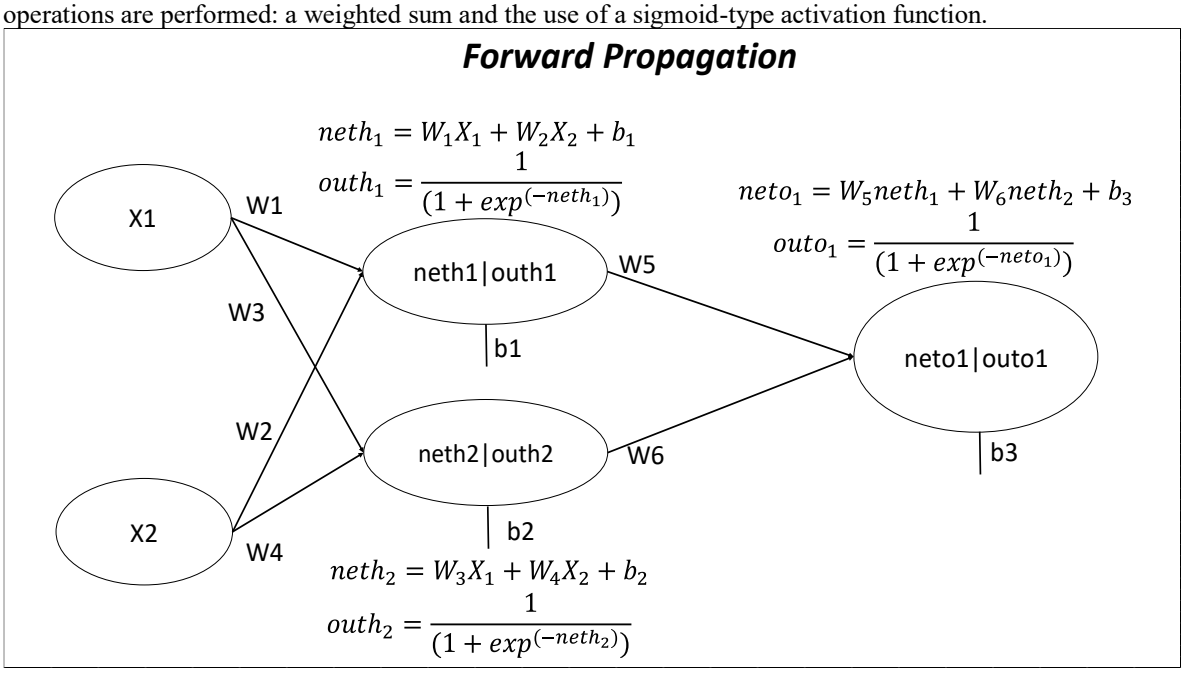


Figure 2. Forward propagation

Figure 3 shows the backward propagation phase. The first parameter to adjust is the bias of the output neuron, which is updated using two gradients: the error gradient of the function and the gradient of the activation function of the output neuron based on the chain rule.

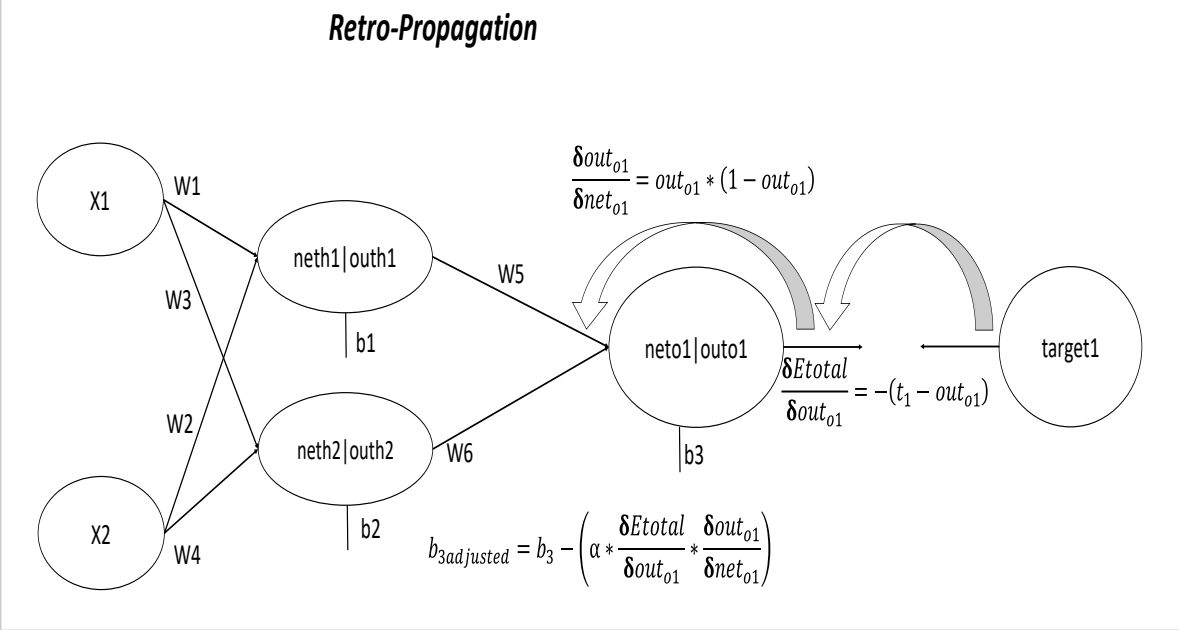


Figure 3. Adjustment of the output layer bias using the backpropagation algorithm.

Figure 4 shows the adjustment of the W5 weight of the output layer using the backpropagation algorithm and the chain rule through three gradients: error gradient, activation function gradient, and weight gradient W5.

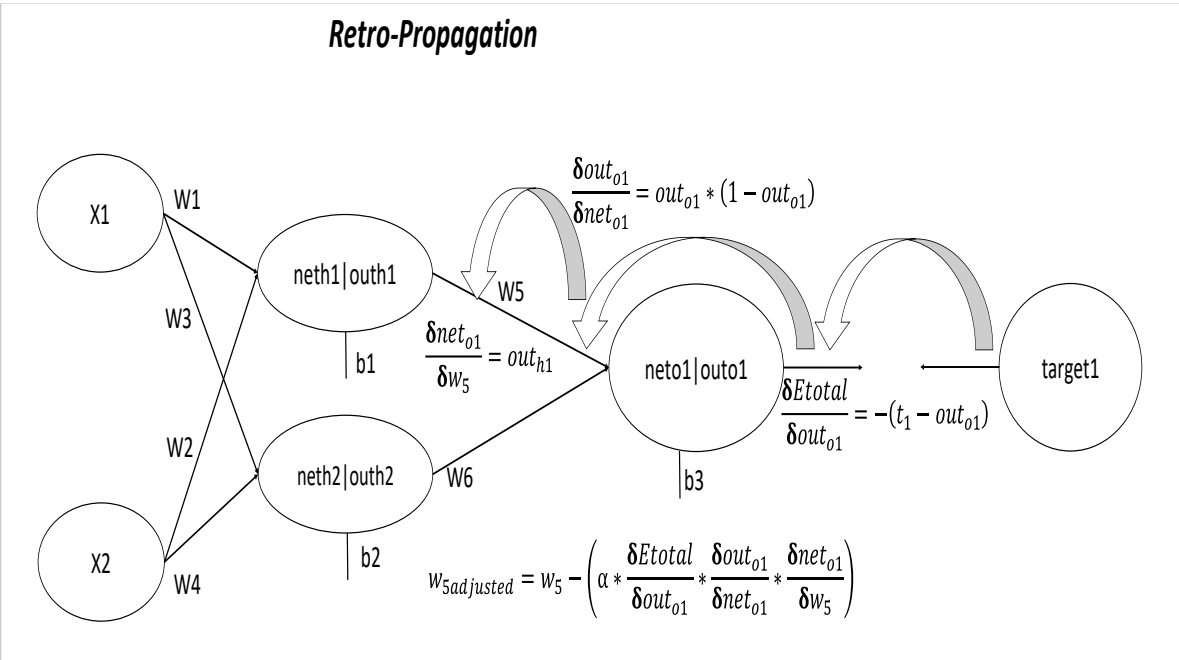


Figure 4. Adjustment of the weight W5 of the output layer using the backpropagation algorithm.

Figure 5 shows how the hidden-layer bias is adjusted using the chain rule. This adjustment uses two gradients: (1) the cumulative error gradient, which includes the error function gradient, activation function gradient, and hidden layer output gradient, and (2) the gradient of the hidden layer neuron's activation function.

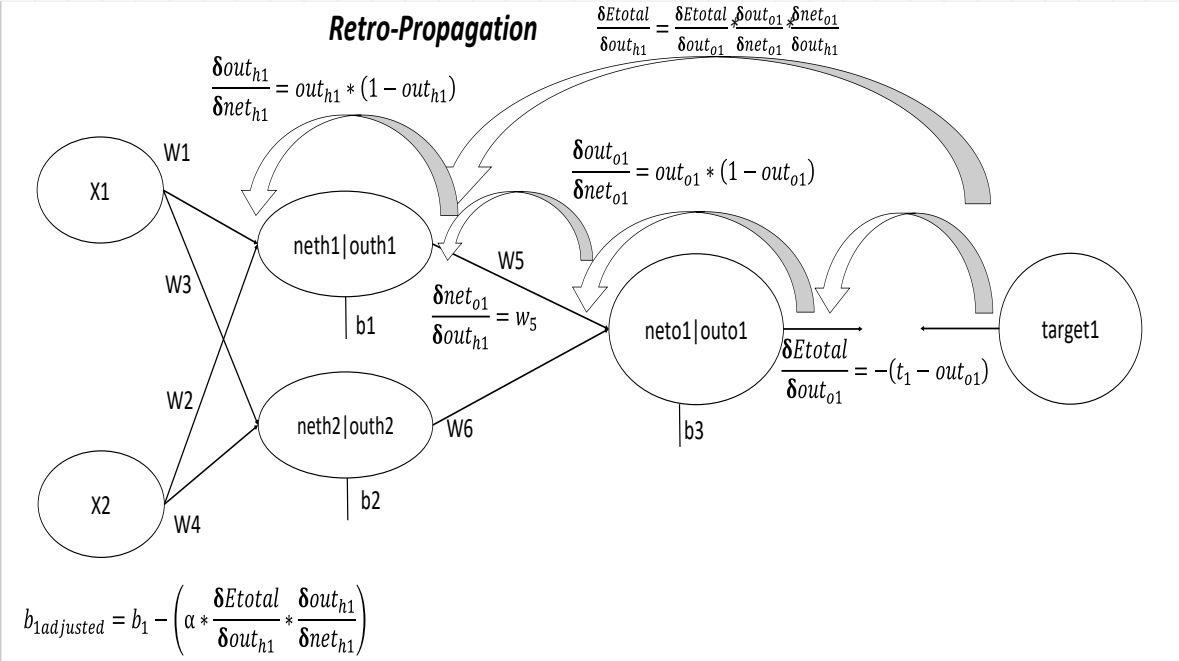


Figure 5. Adjustment of the hidden layer bias using the backpropagation algorithm.

Figure 6 shows how the weight W1 of the hidden-layer is adjusted using the chain rule. This adjustment uses three gradients: (1) the cumulative error gradient, which includes the error function gradient, activation function gradient, and hidden layer output gradient, and (2) the gradient of the hidden layer neuron's activation function and (3) the gradient of the weight W1.

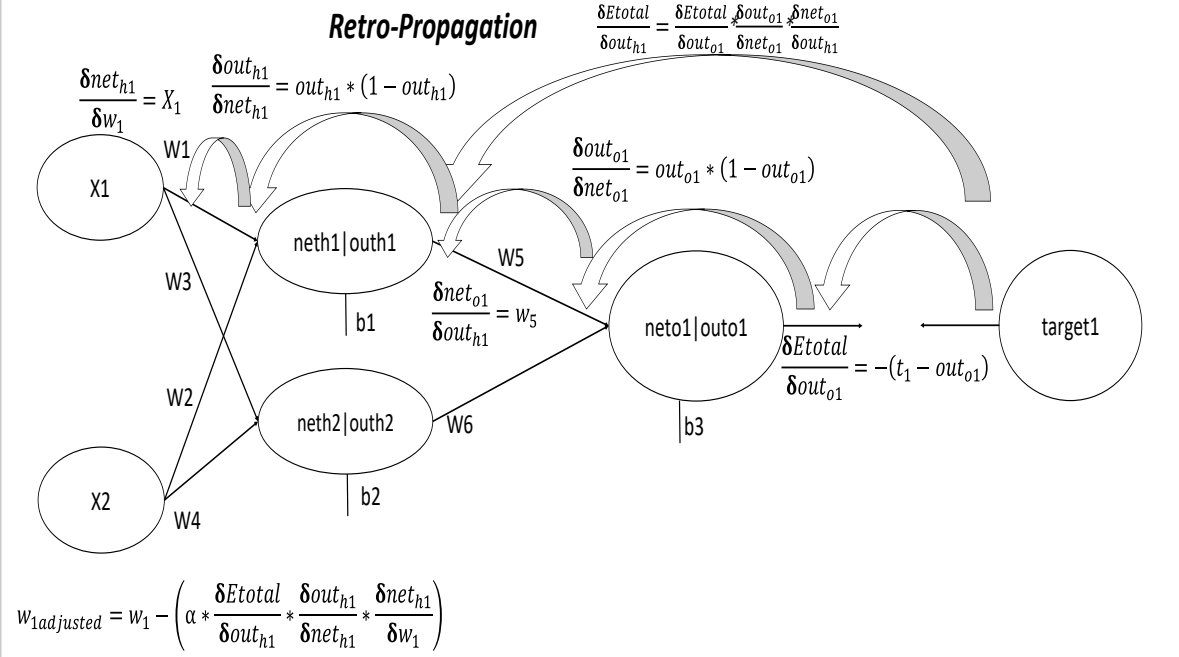


Figure 6. Adjustment of the weight W1 of the hidden layer using the backpropagation algorithm.

3. Results

The first step in training an artificial neural network is data processing, which enables the network to perform better. In Table 1, we observe the data for the input variables and the output variable, along with their preprocessing on a scale of 0 to 1.

Table 1. Input and output variables: Preprocessing.

Pattern	X1 Car Weight	X2 Horsepower	Y1 Kilometer per liter	X1	X2	Y1
1	3504	130	7.65	0.528055342	0.330708661	0.473684211
2	3693	165	6.38	0.600691776	0.606299213	0.315789474
3	3436	150	7.65	0.501921599	0.488188976	0.473684211
4	3433	150	6.80	0.500768640	0.488188976	0.368421053
5	2234	113	11.05	0.039969254	0.196850394	0.894736842
6	2648	90	8.93	0.199077633	0.015748031	0.631578947
7	4615	215	4.25	0.955034589	1	0.052631579
8	4376	200	4.25	0.863182168	0.881889764	0.052631579
9	4732	210	4.68	1.000000000	0.960629921	0.105263158
10	4382	193	3.83	0.865488086	0.826771654	0
11	2130	88	11.48	0.000000000	0	0.947368421
12	2264	90	11.90	0.051498847	0.015748031	1

Propagation of the forward network in four phases. In Phase 1 of forward propagation in the multilayer perceptron neural network, the weights are aggregated with the input variable and the corresponding bias, yielding the net input to the neuron in the hidden layer (neth1). As a phase 2, an activation function of the sigmoidal type is used to obtain the output of each neuron in the hidden layer (out(h1)). See Figure 7. Phase 3 consists of aggregating the two outputs of the hidden layer with their corresponding weights and the bias of the output layer, and determining the net input to the neuron of the output layer (net(o1)). Finally, phase 4 uses a sigmoidal activation function to obtain the neuron's output in the output layer (out(o1)). These calculations are determined in the second row of the Excel spreadsheet. See Figure 8.

	A	B	C	D	E	F	G	H	I	J
1	Input 1 X1 $X_1 = Car\ Weight$	Input 2 X2 $X_2 = Horse\ power$	Weight 1 w1 $w_1 = Random\ (\ )$	Weight 2 w2 $w_2 = Random\ (\ )$	Weight 3 w3 $w_3 = Random\ (\ )$	Weight 4 w4 $w_4 = Random\ (\ )$	Bias 1 b1 $b_1 = Random\ (\ )$	Bias 2 b2 $b_2 = Random\ (\ )$	Phase 1: neth1 $(X_1 * w_1) + (X_2 * w_2) + b_1$ *Cells* $(A_2 * C_2) + (B_2 * D_2) + G_2$	Phase 2: outh1 $\frac{1}{(1+exp^{(-net_{h1})})}$ $\frac{1}{(1+exp^{(-I_2)})}$
2	0.528055342	0.330708661	0.554534261	0.048047783	0.678563585	0.862271786	0.995034605	0.409521802	1.303749202	0.786465291
3	0.600691776	0.606299213								
4	0.501921599	0.488188976								
5	0.50076864	0.488188976								
6	0.039969254	0.196850394								
7	0.199077633	0.015748031								
8	0.955034589	1								
9	0.863182168	0.881889764								
10	1	0.960629921								
11	0.865488086	0.826771654								
12	0	0								
13	0.051498847	0.015748031								

Figure 7. Feedforward propagation hidden layer: First iteration, line2.

	K	L	M	N	O	P	Q	R
1	Phase 1: neth2 $(X_1 * w_1) + (X_2 * w_2) + b_1$ *Cells* $(A_2 * E_2) + (B_2 * F_2) + H_2$	Phase 2: outh2 $\frac{1}{(1+exp^{(-net_{h2})})}$ $\frac{1}{(1+exp^{(-K_2)})}$	Weight 5 w5 $w_5 = Random\ (\ )$	Weight 6 w6 $w_6 = Random\ (\ )$	Bias 3 b3 $b_3 = Random\ (\ )$	Phase 3: neto1 $(out_{h1} * w_5) + (out_{h2} * w_6) + b_3$ *Cells* $(J_2 * M_2) + (l_2 * N_2) + O_2$	Phase 4: outo1 $\frac{1}{(1+exp^{(-net_{o1})})}$ $\frac{1}{(1+exp^{(-P_2)})}$	Target
2	0.947695181	0.720651424	0.75527405	0.184706511	0.853268642	1.537217137	0.823059815	0.47368421
3								0.31578947
4								0.47368421
5								0.36842105
6								0.89473684
7								0.63157895
8								0.05263158
9								0.05263158
10								0.10526316
11								0
12								0.94736842
13								1

Figure 8. Feedforward propagation output layer: First iteration, line2.

In the following columns, the gradients required for the training of the network structure 2-2-1 are calculated. First, the gradient of the error, the gradient of the activation function of the output neuron, and the gradients of the output-layer weights are calculated. Subsequently, the cumulative error gradient for the hidden-layer neurons, the gradients of the hidden-layer activation functions, and the gradients of the hidden-layer weights are calculated. In row 1, the description of the necessary gradients is provided. See Figures (9-10).

	S	T	U	V	W	X	Y	Z
1	Error gradient $\frac{\delta Etotal}{\delta out_{o1}} = -(t_1 - out_{o1})$ Cells $\frac{\delta Etotal}{\delta out_{o1}} = -(R_2 - Q_2)$	Activation 1 Gradient $\frac{\delta out_{o1}}{\delta net_{o1}} = out_{o1} * (1 - out_{o1})$ Cells $\frac{\delta out_{o1}}{\delta net_{o1}} = Q_2 * (1 - Q_2)$	*Weight W5 Gradient* $\frac{\delta net_{o1}}{\delta w_5} = out_{h1}$ Cells $\frac{\delta net_{o1}}{\delta w_5} = J_2$	*Weight W6 Gradient* $\frac{\delta net_{o1}}{\delta w_6} = out_{h2}$ Cells $\frac{\delta net_{o1}}{\delta w_6} = L_2$	outh1 gradient $\frac{\delta net_{o1}}{\delta out_{h1}} = w_5$ Cells $\frac{\delta net_{o1}}{\delta out_{h1}} = M_2$	outh2 gradient $\frac{\delta net_{o1}}{\delta out_{h2}} = w_6$ Cells $\frac{\delta net_{o1}}{\delta out_{h2}} = N_2$	*Accumulated error Outh1 Gradient* $\frac{\delta Etotal}{\delta out_{h1}} = \frac{\delta Etotal}{\delta out_{o1}} \frac{\delta out_{o1}}{\delta net_{o1}} \frac{\delta net_{o1}}{\delta out_{h1}}$ Cells $\frac{\delta Etotal}{\delta out_{h1}} = S_2 * T_2 * W_2$	*Accumulated error Outh2 Gradient* $\frac{\delta Etotal}{\delta out_{h2}} = \frac{\delta Etotal}{\delta out_{o1}} \frac{\delta out_{o1}}{\delta net_{o1}} \frac{\delta net_{o1}}{\delta out_{h2}}$ Cells $\frac{\delta Etotal}{\delta out_{h2}} = S_2 * T_2 * X_2$
2	0.375106628	0.128344951	0.616593977	0.669178206	0.877514318	0.54917643	0.042246209	0.026439024
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								

Figure 9. Gradients: First iteration, line2.

	AA	AB	AC	AD	AE	AF	AG
1	Activation 2 Gradient $\frac{\delta out_{h1}}{\delta net_{h1}} = out_{h1} * (1 - out_{h1})$ Cells $\frac{\delta out_{h1}}{\delta net_{h1}} = J_2 * (1 - J_2)$	Activation 3 Gradient $\frac{\delta out_{h2}}{\delta net_{h2}} = out_{h2} * (1 - out_{h2})$ Cells $\frac{\delta out_{h2}}{\delta net_{h2}} = L_2 * (1 - L_2)$	*Weight W1 Gradient* $\frac{\delta net_{h1}}{\delta w_1} = X_1$ Cells $\frac{\delta net_{h1}}{\delta w_1} = A_2$	*Weight W2 Gradient* $\frac{\delta net_{h1}}{\delta w_2} = X_2$ Cells $\frac{\delta net_{h1}}{\delta w_2} = B_2$	*Weight W3 Gradient* $\frac{\delta net_{h2}}{\delta w_3} = X_1$ Cells $\frac{\delta net_{h2}}{\delta w_3} = A_2$	*Weight W4 Gradient* $\frac{\delta net_{h2}}{\delta w_4} = X_2$ Cells $\frac{\delta net_{h2}}{\delta w_4} = B_2$	α Learning coefficient
2	0.236405844	0.221378735	0.528055342	0.330708661	0.528055342	0.330708661	0.25
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							

Figure 10. Gradients: First iteration, line2.

In the third row, the adjustments to the hidden-layer weights and bias are determined using the gradients obtained in row 2. The formulas used are shown in Figures 11 and 12. Phase B: For the bias adjustment of the one hidden-layer neuron, two gradients are used: the gradient of the accumulated error in the hidden neuron 1 and the gradient of the activation function of the hidden neuron 1. For adjusting the weights of hidden neuron 1, three gradients are used: the cumulative error gradient in hidden neuron 1, the gradient of the hidden neuron activation function 1, and the gradients of the weights w1 and w2. For the bias adjustment of the two hidden-layer neurons, two gradients are used: the gradient of the accumulated error in hidden neuron 2 and the gradient of the activation function of hidden neuron 2. For the adjustment of the weights of the hidden neuron 2, three gradients are used: the accumulated error gradient in hidden neuron 2, the gradient of the hidden neuron activation function 2, and the gradient of the weights w3 and w4.

	C	D	E
1	Weight 1 w1 *Line* 1 $w_1 = Random\ (\ )$ ***Line* 2** $w_{1adjusted} = w_1 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h1}} * \frac{\delta out_{h1}}{\delta net_{h1}} * \frac{\delta net_{h1}}{\delta w_1}\right)$ $C_3 = C_2 - (AG_2 * Y_2 * AA_2 * AC_2)$	Weight 2 w2 *Line* 1 $w_2 = Random\ (\ )$ ***Line* 2** $w_{2adjusted} = w_2 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h1}} * \frac{\delta out_{h1}}{\delta net_{h1}} * \frac{\delta net_{h1}}{\delta w_2}\right)$ $D_3 = D_2 - (AG_2 * Y_2 * AA_2 * AD_2)$	Weight 3 w3 *Line* 1 $w_3 = Random\ (\ )$ ***Line* 2** $w_{3adjusted} = w_3 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h2}} * \frac{\delta out_{h2}}{\delta net_{h2}} * \frac{\delta net_{h2}}{\delta w_2}\right)$ $E_3 = E_2 - (AG_2 * Z_2 * AB_2 * AE_2)$
2	0.047422134	0.843090443	0.707038529
3	0.046638763	0.842599836	0.705944377
4			
5			
6			
7			

Figure 11. Backpropagation: Second iteration, line3.

	F	G	H
1	Weight 4 w4 *Line* 1 $w_4 = Random\ (\ )$ ***Line* 2** $w_{4adjusted} = w_4 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h2}} * \frac{\delta out_{h2}}{\delta net_{h2}} * \frac{\delta net_{h2}}{\delta w_4}\right)$ $F_3 = F_2 - (AG_2 * Z_2 * AB_2 * AF_2)$	Bias 1 b1 *Line* 1 $b_1 = Random\ (\ )$ ***Line* 2** $b_{1adjusted} = b_1 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h1}} * \frac{\delta out_{h1}}{\delta net_{h1}}\right)$ $G_3 = G_2 - (AG_2 * Y_2 * AA_2)$	Bias 2 b2 *Line* 1 $b_2 = Random\ (\ )$ ***Line* 2** $b_{2adjusted} = b_2 - \left(\alpha * \frac{\delta Etotal}{\delta out_{h2}} * \frac{\delta out_{h2}}{\delta net_{h2}}\right)$ $H_3 = H_2 - (AG_2 * Z_2 * AB_2)$
2	0.13468847	0.237948175	0.537761039
3	0.134003228	0.236464674	0.535688998
4			
5			
6			
7			

Figure 12. Backpropagation: Second iteration, line3.

In row 3, the adjustments to the weights and bias of the output layer are determined, using the gradients obtained in row 2. The formulas used are shown in row 1.Phase A: For the bias adjustment of the one output layer neuron, two gradients are used: the gradient of the error in the output neuron 1 and the gradient of the activation function of the output neuron 1. For adjusting the weights of output neuron 1, three gradients are used: the gradient of the error in output neuron 1, the gradient of the output neuron activation function 1, and the gradients of the weights w5 and w6. See Figures 13-17.

	M	N	O
	Weight 5 w5 *Line* 1 $w_5 = Random\ (\ )$ ***Line* 2** $w_{5adjusted} = w_5 - \left(\alpha * \frac{\delta Etotal}{\delta out_{o1}} * \frac{\delta out_{o1}}{\delta net_{o1}} * \frac{\delta net_{o1}}{\delta w_5}\right)$ $M_3 = M_2 - (AG_2 * S_2 * T_2 * U_2)$	Weight 6 w6 *Line* 1 $w_6 = Random\ (\ )$ ***Line* 2** $w_{6adjusted} = w_6 - \left(\alpha * \frac{\delta Etotal}{\delta out_{o1}} * \frac{\delta out_{o1}}{\delta net_{o1}} * \frac{\delta net_{o1}}{\delta w_6}\right)$ $N_3 = N_2 - (AG_2 * S_2 * T_2 * V_2)$	Bias 3 b3 *Line* 1 $b_3 = Random\ (\ )$ ***Line* 2** $b_{3adjusted} = b_3 - \left(\alpha * \frac{\delta Etotal}{\delta out_{o1}} * \frac{\delta out_{o1}}{\delta net_{o1}}\right)$ $O_3 = O_2 - (AG_2 * S_2 * T_2)$
1			
2	0.54101041	0.875843129	0.80526517
3	0.533554355	0.867325459	0.793471956
4			
5			
6			
7			

Figure 13. Backpropagation: Second iteration, line3

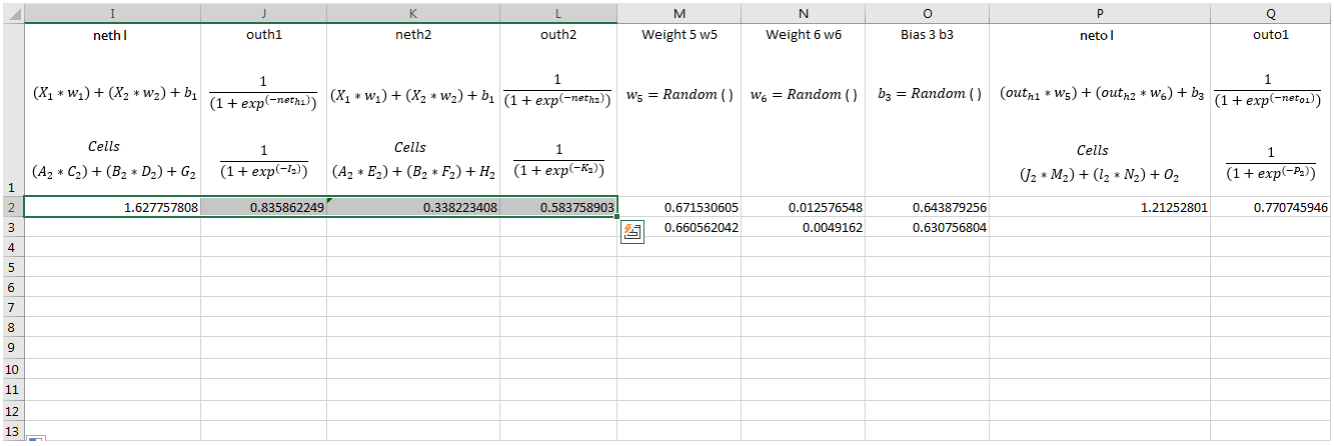
The next step is to select the cells with the weight and bias settings, then drag them to the number of iterations specified for learning the network. In this example, 5000 iterations were used; that is, the data was dragged up to line 5000. In the same way, the forward-propagation cells and gradients in row 2 are selected and dragged to the specified number of iterations. See Figure 6.

	A	B	C	D	E	F	G	H	I	J
	Input 1 X1 $X_1 = Car\ Weight$	Input 2 X2 $X_2 = Horse\ power$	Weight 1 w1 $w_1 = Random\ (\ )$	Weight 2 w2 $w_2 = Random\ (\ )$	Weight 3 w3 $w_3 = Random\ (\ )$	Weight 4 w4 $w_4 = Random\ (\ )$	Bias 1 b1 $b_1 = Random\ (\ )$	Bias 2 b2 $b_2 = Random\ (\ )$	neth l $(X_1 * w_1) + (X_2 * w_2) + b_1$ *Cells* $(A_2 * C_2) + (B_2 * D_2) + G_2$	outh1 $\frac{1}{(1 + exp^{(-neth_1)})}$ $\frac{1}{(1 + exp^{(-I_2)})}$
1										
2	0.528055342	0.330708661	0.105488666	0.812822265	0.193858921	0.170782	0.673045442	0.455957919	0.997556659	0.730577917
3	0.600691776	0.606299213	0.105470617	0.812810962	0.193657878	0.170656092	0.673011261	0.455577197		
4	0.501921599	0.488188976								
5	0.50076864	0.488188976								
6	0.039969254	0.196850394								
7	0.199077633	0.015748031								
8	0.955034589	1								
9	0.863182168	0.881889764								
10	1	0.960629921								
11	0.865488086	0.826771654								
12	0	0								
13	0.051498847	0.015748031								

Figure 14. Select and drag until 5000 iterations.

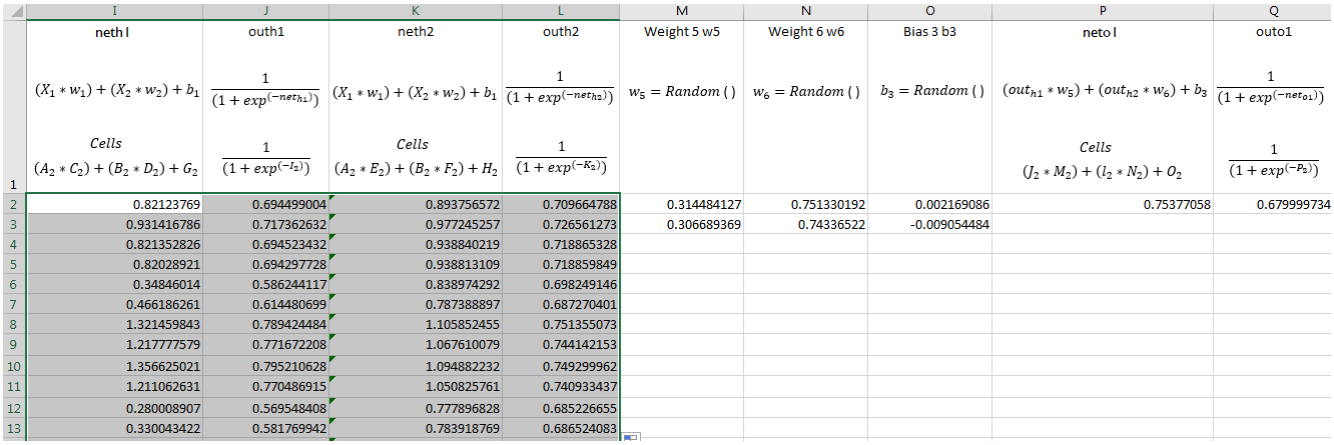
	A	B	C	D	E	F	G	H	I	J
	Input 1 X1 $X_1 = Car\ Weight$	Input 2 X2 $X_2 = Horse\ power$	Weight 1 w1 $w_1 = Random\ (\ )$	Weight 2 w2 $w_2 = Random\ (\ )$	Weight 3 w3 $w_3 = Random\ (\ )$	Weight 4 w4 $w_4 = Random\ (\ )$	Bias 1 b1 $b_1 = Random\ (\ )$	Bias 2 b2 $b_2 = Random\ (\ )$	neth l $(X_1 * w_1) + (X_2 * w_2) + b_1$ *Cells* $(A_2 * C_2) + (B_2 * D_2) + G_2$	outh1 $\frac{1}{(1 + exp^{(-neth_1)})}$ $\frac{1}{(1 + exp^{(-I_2)})}$
1										
2	0.528055342	0.330708661	0.795110252	0.661059397	0.234565636	0.32657291	0.989277524	0.106359281	1.627757808	0.835862249
3	0.600691776	0.606299213	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
4	0.501921599	0.488188976	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
5	0.50076864	0.488188976	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
6	0.039969254	0.196850394	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
7	0.199077633	0.015748031	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
8	0.955034589	1	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
9	0.863182168	0.881889764	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
10	1	0.960629921	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
11	0.865488086	0.826771654	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
12	0	0	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		
13	0.051498847	0.015748031	0.794471836	0.660659572	0.234544461	0.326559648	0.98806853	0.10631918		

Figure 15. Select and drag until 5000 iterations.



	I	J	K	L	M	N	O	P	Q
	neth l	outh1	neth2	outh2	Weight 5 w5	Weight 6 w6	Bias 3 b3	neto l	outo1
	$(X_1 * w_1) + (X_2 * w_2) + b_1$	$\frac{1}{(1+exp^{(-net_{h1})})}$	$(X_1 * w_1) + (X_2 * w_2) + b_1$	$\frac{1}{(1+exp^{(-net_{h2})})}$	$w_5 = Random\ ()$	$w_6 = Random\ ()$	$b_3 = Random\ ()$	$(out_{h1} * w_5) + (out_{h2} * w_6) + b_3$	$\frac{1}{(1+exp^{(-net_{o1})})}$
	Cells $(A_2 * C_2) + (B_2 * D_2) + G_2$	$\frac{1}{(1+exp^{(-I_2)})}$	*Cells* $(A_2 * E_2) + (B_2 * F_2) + H_2$	$\frac{1}{(1+exp^{(-K_2)})}$				*Cells* $(J_2 * M_2) + (l_2 * N_2) + O_2$	$\frac{1}{(1+exp^{(-P_2)})}$
1									
2	1.627757808	0.835862249	0.338223408	0.583758903	0.671530605	0.012576548	0.643879256	1.21252801	0.770745946
3					0.660562042	0.0049162	0.630756804		
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									

Figure 16. Select and drag until 5000 iterations.



	I	J	K	L	M	N	O	P	Q
	neth l	outh1	neth2	outh2	Weight 5 w5	Weight 6 w6	Bias 3 b3	neto l	outo1
	$(X_1 * w_1) + (X_2 * w_2) + b_1$	$\frac{1}{(1+exp^{(-net_{h1})})}$	$(X_1 * w_1) + (X_2 * w_2) + b_1$	$\frac{1}{(1+exp^{(-net_{h2})})}$	$w_5 = Random\ ()$	$w_6 = Random\ ()$	$b_3 = Random\ ()$	$(out_{h1} * w_5) + (out_{h2} * w_6) + b_3$	$\frac{1}{(1+exp^{(-net_{o1})})}$
	Cells $(A_2 * C_2) + (B_2 * D_2) + G_2$	$\frac{1}{(1+exp^{(-I_2)})}$	*Cells* $(A_2 * E_2) + (B_2 * F_2) + H_2$	$\frac{1}{(1+exp^{(-K_2)})}$				*Cells* $(J_2 * M_2) + (l_2 * N_2) + O_2$	$\frac{1}{(1+exp^{(-P_2)})}$
1									
2	0.82123769	0.694499004	0.893756572	0.709664788	0.314484127	0.751330192	0.002169086	0.75377058	0.679999734
3	0.931416786	0.717362632	0.977245257	0.726561273	0.306689369	0.74336522	-0.009054484		
4	0.821352826	0.694523432	0.938840219	0.718865328					
5	0.82028921	0.694297728	0.938813109	0.718859849					
6	0.34846014	0.586244117	0.838974292	0.698249146					
7	0.466186261	0.614480699	0.787388897	0.687270401					
8	1.321459843	0.789424484	1.105852455	0.751355073					
9	1.217777579	0.771672208	1.067610079	0.744142153					
10	1.356625021	0.795210628	1.094882232	0.749299962					
11	1.211062631	0.770486915	1.050825761	0.740933437					
12	0.280008907	0.569548408	0.777896828	0.685226655					
13	0.330043422	0.581769942	0.783918769	0.686524083					

Figure 17. Select and drag until 5000 iterations.

Figure 18 shows the neural network forecast in an Excel sheet, based on correctly filling rows 2 and 3, selected and dragged to 5000 iterations, in a simple, straightforward, and easy way. The results are very reliable, with a low mean-squared error. See Table 2.

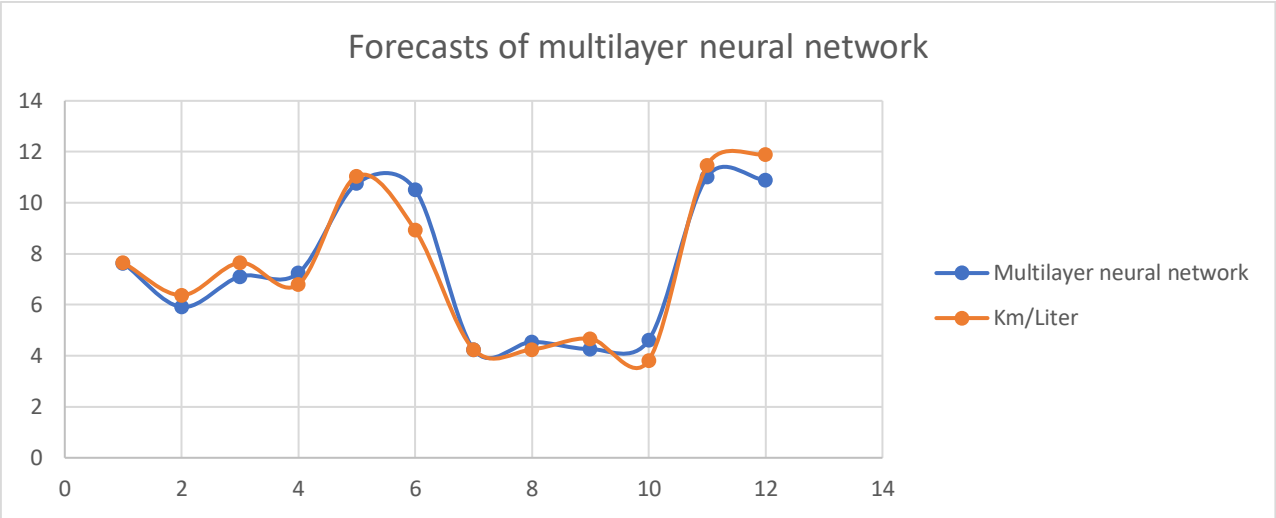


Figure 18. Forecasts of multilayer neural network.

Table 2. Comparative of neuronal network output and targets

OO1 Neural network	Transformed	Originals	MSE
0.472385039	7.640462469	7.65	5.50423E-05
0.258847746	5.915933662	6.38	0.10573645
0.406972518	7.112190445	7.65	0.1451334
0.423614309	7.246589676	6.80	0.099342589
0.858747014	10.76072472	11.05	0.04223991
0.828131961	10.51347732	8.93	1.259861656
0.052585401	4.250157393	4.25	6.95395E-08
0.089418161	4.547619042	4.25	0.044130853
0.054554789	4.266062187	4.68	0.083853795
0.098713847	4.622691077	3.83	0.317774908
0.891878179	11.02829226	11.48	0.100414559
0.873929719	10.88334037	11.90	0.518309158
		Average	0.226404366

Table 3 shows another application of a neural network with a 2-5-1 topology, 2 input variables, 5 hidden neurons, and 1 output neuron. The inbound variables represent two-day weekly sales at a restaurant, and the outbound variable represents another day of the week's sales. The neural network aims to estimate the sales of any given day based on information from two previous days. The table shows actual sales data and pre-processing of data on a scale from 0 to 1. The results are very reliable, with a low mean-squared error. See Table 4.

Table 3. Input and output variables for product sales: Preprocessing.

Pattern	X1 Thursday sales	X2 Friday sales	Y1 Saturday sales	X1	X2	Y1
1	$ 10,657.00	$ 6,184.00	$ 10,657.00	0.80557021	0.39807858	0
2	$ 5,554.00	$ 7,361.00	$ 5,554.00	0.26722228	0.54304717	0.72911531
3	$ 9,106.00	$ 6,030.00	$ 9,106.00	0.64194535	0.37911073	0.68803068
4	$ 8,436.00	$ 7,642.00	$ 8,436.00	0.57126279	0.57765735	1
5	$ 6,954.00	$ 7,799.00	$ 6,954.00	0.41491719	0.5969947	0.18666119

6	$ 12,500.00	$ 7,890.00	$ 12,500.00	1	0.60820298	0.44261846
7	$ 8,706.00	$ 11,071.00	$ 8,706.00	0.59974681	1	0.07751301
8	$ 9,503.00	$ 5,583.00	$ 9,503.00	0.68382741	0.32405469	0.31238017
9	$ 4,232.00	$ 6,078.00	$ 4,232.00	0.12775609	0.38502279	0.30197206
10	$ 3,731.00	$ 8,078.00	$ 3,731.00	0.07490242	0.63135854	0.08121063
11	$ 9,114.00	$ 6,641.00	$ 9,114.00	0.64278932	0.4543663	0.26376335
12	$ 5,037.00	$ 2,952.00	$ 5,037.00	0.21268066	0	0.86565325
13	$ 3,021.00	$ 4,647.00	$ 3,021.00	0	0.20876955	0.66077787
14	$ 7,701.00	$ 7,746.00	$ 7,701.00	0.49372297	0.59046681	0.46946042
15	$ 7,802.00	$ 6,159.00	$ 7,802.00	0.5043781	0.39499938	0.46329773

Figure 19 shows the neural network forecast in an Excel sheet, based on correctly filling rows 2 and 3, selected and dragged to 5000 iterations, in a simple, straightforward, and easy way. The results are very reliable, with a low mean-squared error. See Table 4.

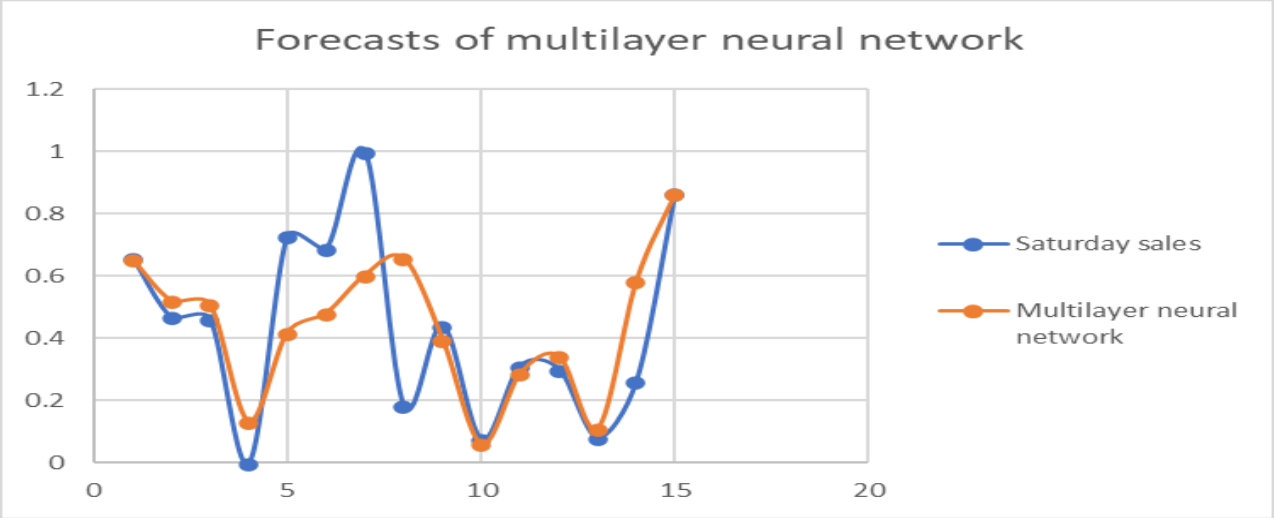


Figure 19. Forecasts of multilayer neural network.

Table 4. Comparative of neuronal network output and targets

OO1 Neural network	Transformed	Originals	Absolute error
0.656906393	$ 6,513.73	$ 6,542.00	$ 28.27
0.469465142	$ 5,145.03	$ 5,145.00	$ 0.03
0.464169723	$ 5,106.37	$ 5,100.00	$ 6.37
0.116636324	$ 2,568.68	$ 1,717.00	$ 851.68
0.388428409	$ 4,553.30	$ 7,041.00	$ 2,487.70
0.505519902	$ 5,408.31	$ 6,741.00	$ 1,332.69
0.674378111	$ 6,641.31	$ 9,019.00	$ 2,377.69
0.753384189	$ 7,218.21	$ 3,080.00	$ 4,138.21
0.417003494	$ 4,761.96	$ 4,949.00	$ 187.04
0.066514281	$ 2,202.69	$ 2,283.00	$ 80.31
0.300398447	$ 3,910.51	$ 3,998.00	$ 87.49
0.33379663	$ 4,154.38	$ 3,922.00	$ 232.38
0.114763949	$ 2,555.01	$ 2,310.00	$ 245.01
0.598216854	$ 6,085.18	$ 3,643.00	$ 2,442.18

0.867430981	$ 8,050.98	$ 8,038.00	$ 12.98
		Average	$ 967.34

Finally, Tables 5-6 present the results of scientific products obtained with undergraduate and graduate students who did not possess programming knowledge and developed neural network applications using the proposed methodology for developing a multilayer neural network in an Excel sheet. Eleven articles have now been published in peer-reviewed journals and at national or international congresses. Three students obtained a bachelor's degree and four a master's degree, all of whom used the methodology proposed in this article.

Table 5. Articles published using the proposed methodology (Spanish).

Paper Title	Authors	Journal/congress	Year	Country
Implementation of hybrid radial basis neural networks for the classification of customers of an SME	Bedolla-Guzmán Jonathan, Baeza-Serrato Roberto	*Journal of Multidisciplinary Engineering Science Studies*	2019	Germany
Sales Prediction For An SME Using An Artificial Neural Network	Martínez-López Mariana, Baeza-Serrato Roberto	*Journal of Multidisciplinary Engineering Science Studies*	2022	Germany
Redes neuronales artificiales para la clasificación de ventas en una tienda de abarrotes	Álvarez-Niño E., Baeza-Serrato Roberto	*Congreso Internacional de logística y cadena de suministro CiLOG 2018*	2018	México
Desarrollo de una red neuronal artificial para la clasificación de proveedores de una Pyme del Sur de Guanajuato.	Guzmán-Torres N., Baeza-Serrato Roberto.	*Congreso Internacional de logística y cadena de suministro CiLOG 2018*	2018	México
Diseño y desarrollo de una red neuronal tipo perceptrón multicapa en Excel para la clasificación y selección de proveedores en las tiendas de conveniencia de Yuriria, Guanajuato	Acevedo-Arcila I. A, Baeza-Serrato Roberto	*Academia Journals Morelia 2019*	2019	Morelia, México
Desarrollo de una red neuronal multicapa para realizar un pronóstico de ventas en una pyme del Sur de Guanajuato	Lara-Alvarado S. L, Baeza-Serrato Roberto.	*Academia Journals Morelia 2019*	2019	Morelia, México
Diseño y desarrollo de una red de base radial para clasificar en tres zonas geográficas a los principales clientes de la empresa "El Barista"	Bedolla-Guzmán J. Baeza-Serrato Roberto	*Academia Journals Morelia 2019*	2019	Morelia, México
Comparación de la red neuronal perceptron multicapa y red neuronal de base radial: modelo de predicción y clasificación.	Silva-Núñez J. M, Baeza-Serrato Roberto.	*Congreso Internacional de logística y cadena de suministro CiLOG 2019.*	2019	Guadalajara, México

Desarrollo de una red multicapa en una MiPyme del sur de Guanajuato.	Sánchez-Alcántar P., Baeza-Serrato Roberto.	*Congreso Internacional de logística y cadena de suministro CiLOG 2019*.	2019	Guadalajara, México
Diseño y desarrollo de una red neuronal multicapa para pronosticar las emisiones de Co2 de la empresa RaSa.	Pito-Soto P., Baeza-Serrato Roberto.	*Congreso Internacional de logística y cadena de suministro CiLOG 2019*.	2019	Guadalajara, México
Diseño y Desarrollo de una Red Neuronal Basada en Datos del Mapa Satelital.	Mireles-Moreno Y., Baeza-Serrato Roberto.	Academia Journals Celaya 2023	2023	Celaya, México

Table 6. Theses developed using the proposed methodology.

Thesis Title	Student/Director	Degree	Year	University/Country
Control de calidad inteligente en el departamento de American Polo S. A de C. V.	María del Carmen Sánchez Silva, Roberto Baeza-Serrato	*Bachelor*	2020	University of Guanajuato, México
Optimización de rutas de mantenimiento: Desarrollo de una red neuronal de base radial	Leticia Damián Ramírez, Roberto Baeza-Serrato	*Master´s*	2021	University of Guanajuato, México
Desarrollo de redes multicapa con diferentes estructuras para la predicción de ventas en una Pyme del sur de Guanajuato.	Paulina Sánchez Alcantar, Roberto Baeza-Serrato	*Bachelor*	2021	University of Guanajuato, México
Diseño y desarrollo de una red neuronal artificial para la clasificación de proveedores en una industria cartonera	Iván Andrés Arcila Acevedo, Roberto Baeza-Serrato	*Master´s*	2021	University of Guanajuato, México
Predicción de ventas utilizando una red neuronal artificial.	Mariana Martínez López, Roberto Baeza-Serrato	*Bachelor*	2022	University of Guanajuato, México
Diseño y desarrollo de una red neuronal basada en datos del mapa satelital.	Yareli Mireles Moreno, Roberto Baeza-Serrato	*Master´s*	2025	University of Guanajuato, México
Diseño y desarrollo de redes neuronales de base radial para la clasificación de los clientes de una PYME del sur de Guanajuato.	Jonathan Bedolla Guzmán, Roberto Baeza-Serrato	*Master´s*	2025	University of Guanajuato, México

4. Discussion

The proposal of developing a multilayer neural network in an Excel sheet with only two lines allows for the design of any necessary network topology. However, it is very important to understand how forward and backward propagation are performed so that operations and gradients are computed correctly. The more complex the network topology, the more calculations in neurons and gradients will be needed. in the present research a topology is shown that has only one output neuron, one of the limitations of the proposal is that it is not shown how to calculate the cumulative function error when more than one neuron is in the output layer, which in future works will be presented so that students and professionals who do not have knowledge in advanced programming can develop any network topology, however complex it may be, with the proposed method of using only two rows and then dragging up to the number of iterations needed to train the neural network.

5. Conclusions

This research proposes designing a neural network in only two rows of a spreadsheet in Excel for students who do not know advanced programming.

In columns A and B, the 12 training patterns are placed. From column C in row 2, the bias and weights are determined randomly, and the aggregation and activation functions of the two neurons in the hidden layer are calculated. The bias and weights of the hidden layer are determined randomly, along with the corresponding aggregation and activation, and the training patterns of the output variable are used as targets. In row 2, the gradients of the error, the activation gradient, and the output-layer weights are determined. Subsequently, the cumulative gradients of the error in each neuron of the hidden layer are determined, as well as the gradients of the activation function and the gradients of the hidden layer weights.

In row 3, the weights and biases of the output and hidden layers are adjusted using the gradients computed in row 2. Once rows 2 and 3 are filled in, they are selected and dragged to the desired number of iterations, yielding a neural network easily and clearly, without advanced programming.

The main contribution of this research is to provide students in any career with a simple tool for high-impact applications such as prediction, classification, or pattern recognition.

Future work is to develop fuzzy systems and genetic algorithms in Excel spreadsheets to facilitate students or professionals who do not have advanced programming knowledge in using these tools of artificial intelligence.

References

Bautista-Capetillo, C., Robles Rovelo, C. O., González-Trinidad, J., Pineda-Martínez, H., Júnez-Ferreira, H. E., & García-Bandala, M. (2023). Teaching sprinkler irrigation engineering by a spreadsheet tool. *Water, 15*(9), Article 1685. https://doi.org/10.3390/w15091685

Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing, 275*, 278–287. https://doi.org/10.1016/j.neucom.2017.08.040

Chen, J., Yin, H., Zhang, K., Ren, Y., & Zeng, H. (2025). Integration of neural networks in brain–computer interface applications: Research frontiers and trend analysis based on Python. *Engineering Applications of Artificial Intelligence, 151*, Article 110654. https://doi.org/10.1016/j.engappai.2025.110654

Faria, A., & Miranda, G. L. (2026). Operationalising CTT and IRT in spreadsheets: A methodological demonstration for classroom assessment. *Analytics, 5*(1), Article 12. https://doi.org/10.3390/analytics5010012

Guerriero, V. (2023). Maximum likelihood instead of least squares in fracture analysis by means of a simple Excel sheet with VBA macro. *Geosciences, 13*(12), Article 379. https://doi.org/10.3390/geosciences13120379

Guo, R., Li, R., Ding, J., Wen, L., Zhang, Y., Wen, C., & Jiao, M. (2026). Neural network for gas recognition based on MOS electronic nose: Algorithm design and hardware deployment. *Microchemical Journal, 225*, Article 118008. https://doi.org/10.1016/j.microc.2026.118008

Kiran, Y. M., & Srinivas, S. (2021). Simple approach to model photovoltaic module using Solver utility of Microsoft Excel. In *2021 IEEE International Power and Renewable Energy Conference (IPRECON)* (pp. 1–5). IEEE. https://doi.org/10.1109/IPRECON52453.2021.9640780

Lee, J., Yoon, Y., & Geem, Z. W. (2025). ANN-Excel: Artificial neural network framework in Excel. *SoftwareX, 32*, Article 102428. https://doi.org/10.1016/j.softx.2025.102428

Li, J. (2025). Entrepreneurial skill augmented neural network (ESANN): A deep learning approach for enhancing entrepreneurial competencies in teachers. *International Journal of Computational Intelligence Systems, 18*, Article 127. https://doi.org/10.1007/s44196-025-00851-2

Liu, J., González-Fernández, D., Castillo-Cara, M., & García-Castro, R. (2025). TINTOlib: A Python library for transforming tabular data into synthetic images for deep neural networks. *SoftwareX, 32*, Article 102444. https://doi.org/10.1016/j.softx.2025.102444

Lou, Y., & Li, F. (2024). Design of an online education student learning status evaluation model based on dual-improved neural networks. *Intelligent Systems with Applications, 22*, Article 200370. https://doi.org/10.1016/j.iswa.2024.200370

Nagy, T., Csernoch, M., & Biró, P. (2021). The comparison of students' self-assessment, gender, and programming-oriented spreadsheet skills. *Education Sciences, 11*(10), Article 590. https://doi.org/10.3390/educsci11100590

Pinto, A. S., Abreu, A., Costa, E., & Paiva, J. (2023). How machine learning (ML) is transforming higher education: A systematic literature review. *Journal of Information Systems Engineering and Management, 8*(2), Article 21168. https://doi.org/10.55267/iadt.07.13227

Sun, X., & Zhang, Y. (2022). An improved BP neural network algorithm for the evaluation system of innovation and entrepreneurship education in colleges and universities. *Mobile Information Systems, 2022*, Article 1007538. https://doi.org/10.1155/2022/1007538

Tian, J. (2025). A practice-oriented computational thinking framework for teaching neural networks to working professionals. *AI, 6*(7), Article 140. https://doi.org/10.3390/ai6070140

White, J. C., & Lucci, F. (2026). ARMADILLO_VolcaNorm: An open-source, macro-enabled VBA for Excel™ spreadsheet for calculation of CIPW norms and IUGS igneous rock classification for volcanic rocks. *Journal of South American Earth Sciences, 174*, Article 106002. https://doi.org/10.1016/j.jsames.2026.106002

Wu, Z., Wang, Y., & Long, X. (2025). Reform path and empirical exploration of visual communication design education based on graph neural network. *Systems and Soft Computing, 7*, Article 200413. https://doi.org/10.1016/j.sasc.2025.200413