IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 3, March, 2026, pp. 699-706
Journal homepage: http://iieta.org/journals/jesa

Design and Implementation of an Artificial-Intelligence-Based Smart Control Unit for an Automatic Clothes Washing Machine

Azzad Bader Saeed[1*], Sabah A. Gitaffa[1], Reem I. Dawai[2]

[1] Electrical Engineering College, University of Technology, Baghdad 35010, Iraq
[2] Middle Technical University, Department of Electronic Technologies, Baghdad 029008, Iraq

Corresponding Author Email: azzad.b.saeed@uotechnology.edu.iq

https://doi.org/10.18280/jesa.590313

Received: 13 January 2026
Revised: 4 March 2026
Accepted: 11 March 2026
Available online: 31 March 2026

Keywords:
Artificial Neural Network, Field-Programmable Gate Array, Mean Square Error, washing machine

ABSTRACT

This paper presents the design and implementation of a proposed smart control unit for an automatic clothes washing machine. The controller is intended to provide two main features: high processing speed and high reliability. To achieve these objectives, a backpropagation neural network (BPNN) is employed as the core decision-making method. The selection of BPNN is motivated by its capability for accurate decision-making as well as fast training and data processing. Owing to rapid data propagation through its network layers with minimal delay, the proposed controller can operate faster than the microcontrollers commonly used in conventional washing machines. In addition, an optimization procedure is incorporated to improve the accuracy of the output decisions and thus strengthen the reliability of the control process. The proposed BPNN controller is first developed and tested in the MATLAB environment and is then converted into a Very High Speed Integrated Circuit Hardware Description Language (VHDL) description for hardware realization. The controller is designed to generate suitable control decisions for the motors and solenoids of the washing machine according to the states of programmable input switches and internal sensors. The software-level test results show a Mean Square Error (MSE) of zero and 100% agreement between the target and actual outputs, indicating the effectiveness of the proposed BPNN model under the tested conditions. Finally, the VHDL design is implemented on a Field-Programmable Gate Array (FPGA) using the ISE Design Suite software package.

1. INTRODUCTION

The automatic clothes washing machine includes an electronic control unit. It is used to control the sequenced functions of the washing process [1]. The main electrical parts of these machines are:(1) a main motor for washing and drying processes, (2) a secondary motor for water draining, (3) a solenoid for water entry control, (4) an electronic control panel [2, 3]. Since 1980, the manufacturers of these machines have used microcontrollers as control units for the washing process. These microcontrollers can be programmed by C, assembly, or machine code language software [4-6]. The construction of these machines is illustrated in Figure 1. The proposed work deals with the controlling process of these types of washing machine.

Although the microcontroller has some suitable features, such as small size, low cost, low power consumption, and is efficient for limited jobs, it offers some considered disadvantages, such as: (1) the processing speed and memory size are limited and not enough if it is used for complex jobs, (2) the programming is tricky, it needs some complex tools and code instructions, (3) it is sensitive to static electricity, (4) it cannot be interfaced to high voltage and power devices [7]. All these disadvantages make the microcontrollers not reliable enough to use as processors for sensitive applications.

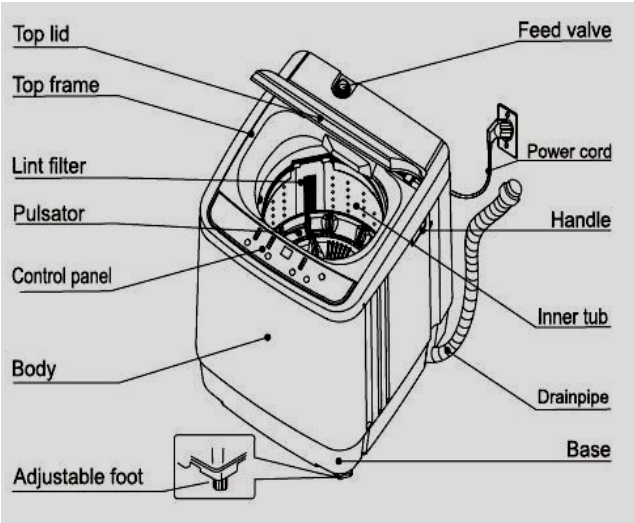

Figure 1. Construction of the clothes washing machine [6]

The backpropagation neural network (BPNN) network was used in the proposed work as an intelligent system to control the processes of the clothes washing machine. The reasons for using this network are: (1) it can present the proper output data

even if an error occurs on the input data, (2) by training this network with several data sets of various washing processes, it can conclude and present more washing process modes. These features were the main reasons of using this network rather than other techniques such as Finite State Machine (FSM), Programmable Logic Controller, or microcontroller.

The first step of the proposed work is designing the software of the proposed controller using the BPNN. Then, in the next step, this simulation is downloaded into a Field-Programmable Gate Array (FPGA). The BPNN is characterized by: its fast training and processing, and a more reliable one of the neural network types, but this network is just a software, and it cannot be compared with the microcontrollers, whereas, the last one is combination of hardware and software, so the BPNN software must be practically implemented by downloading it into fast electrical tool such as FPGA to maintain the 'high speed' feature. As is known, the FPGA is a fast device due to its parallel processing; it can perform big tasks in two or three machine cycles. By implementing the BPNN software into an FPGA, one can make a comparison between this system and the microcontroller, and conclude how this system is faster and more reliable than the microcontroller [8].

The BPNN mechanism depends on presenting an output error signal, which must be minimized as much as possible during the training process. The error signal represents the difference between the actual and desired outputs. Also, the BPNN has an optimization technique to present an accurate decision that supports the fitting between the actual and desired outputs [9]. The BPNN is characterized by: (1) its high speed of learning and processing faster than microcontrollers, (2) ease of installation, updating, and downloading into FPGA, (3) low cost, (4) ease of design due to the use of simple tools and functions in the MATLAB software package [10].

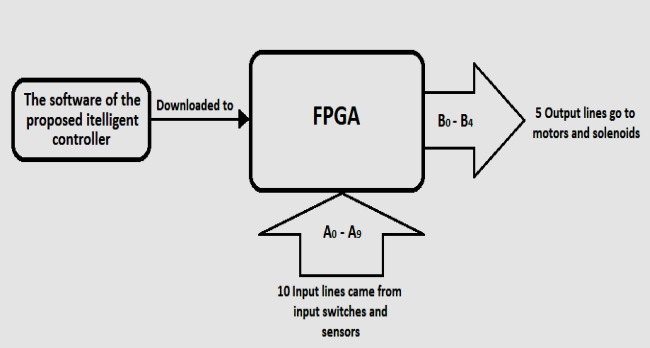

Figure 2. Block diagram of the proposed system

The block diagram for the proposed system is illustrated in Figure 2. As seen, the proposed work starts with the design of the software for the intelligent controller, whereas, the BPNN has been used as the intelligent system for the proposed controller. It contains three layers: input, hidden, and output layers. The input layer involves ten neurons, the hidden layer contains ten neurons, and the output layer consists of five neurons. The ten lines (neurons) of the input layer (A0-A9) are connected to the input switches and sensors of the proposed clothes washing machine, which are arranged as shown in Table 1. The five lines (neurons) of the output layer (B0-B4) are connected to the motors and solenoids of the washing machine, also they are arranged as shown in Table 1. The training function Traingda (training by gradient decent with adaptive learning rate) has been chosen as the learning function for the proposed simulation (software), it is effective and presents excellent results due to high effectiveness of its algorithm with using linear activation function for the hidden and output layers.

The linear functions: Satlin (Bipolar Saturated Linear), and Satlin (Unipolar Saturated Linear) have been chosen as activation functions for the outputs of the hidden and output layers, respectively. The reasons for choosing these activation functions are: (1) their linear relation between the input and output, which can facilitate the conversion of the MATLAB program to Very High Speed Integrated Circuit Hardware Description Language (VHDL) codes, at the same time, reducing the size of the VHDL program; (2) using these activation functions with the training function Traingda can present amazing results.

According to these requirements, the proposed software has been designed using the MATLAB software package, and then this the MATLAB software has been converted to a VHDL program. Then the last one has been downloaded into FPGA. As is known, due to FPGA features, it has been chosen as a suitable device for transforming the proposed software from virtual reality to tangible practical reality. An FPGA is a programmable integrated circuit chip that involves a lot of logic elements: gates, flip-flops, input/output ports, memory, etc. These logic elements are arranged as arrays of Programmable Logic Blocks PLB [11], any digital or logic system can be implemented in an FPGA. The main features of FPGA are: (1) very fast processing because it depends on parallel processing in its operation, where it can complete the job by single or very few clock cycles, (2) low cost and size, (3) ease to use, installation, and upgrading, (3) high reliability for presenting the output (decision) [12, 13].

Table 1. Proposed input and output data set for one of washing process modes

Input Data	Output Data
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9	B_0 B_1 B_2 B_3 B_4
0000000001	00010
0000010001	10000
0000011001	00100
0000001001	00010
0000010001	10000
0000010101	00100
0000000101	01100
0000000011	00000

2. RELATED WORKS

The topic of this work is recent and vital. It touches on using the BPNN as a controller software in various practical applications. In this section, we talk about several related works. These works are recent, and they have touched about using the BPNN as a controller in several applications, which indicates the novelty and importance of the topic of the proposed work.

In 2022, Yan et al. [14] proposed a BPNN network with an optimizing technique of Particle Swarm optimization (PSO) for predicting the risk of symptoms associated with hypertension. The proposed software system of this work has been designed and simulated using the MATLAB software package, which is just a simulation and not implemented in any processing or electronic device.

In 2022, Mohammed et al. [15] developed an improved Artificial Neural Network (ANN) model with an Adaptive

Backpropagation Algorithm (ABPA) for best practice in forecasting long-term load demand of electricity. The proposed ABPA of this work has been used to propose a new formulation of forecasting for adjusting and adapting the values of the forecast. Therefore, the deviation between the future and trained input datasets' for various behaviors has been taken into consideration. This work is also just a simulation system that has been designed using the MATLAB software package and not implemented in any processing device unit.

In 2025, Zhang [16] proposed an algorithm for incremental PID control using the optimizing of the BPNN. He has constructed two architectural models of deep learning and transfer functions. In this work, a proposed BPNN has been constructed from three neurons in the input layer, nine neurons in the hidden layer, and three neurons in the output layer, which has been used for adjusting the parameters of PID for realizing the collaborative optimization of differential, integral, and proportion coefficients. The proposed BPNN is integrated with the incremental PID for realizing the controlling output by the Feed-forward Network Interface (FBI) module. The proposed system of this work is just a simulation and was not implemented in any processing unit.

3. MATERIAL AND METHODS

3.1 Backpropagation neural network backpropagation neural network

BPNN is one of the effective, reliable, fast learning, and fast processing neural networks. It is considered a feedforward network, whereas the input data propagates in a forward direction from the input layer to the output layer through the hidden layer. As a result, an error function will be presented at the output of the output layer. This error is formulated and multiplied by several parameters to produce an updating signal that propagates backward to update the weights of the input and hidden layer. This process is repeated several times to minimize the error function as much as possible to make the target output data approach or fit the real data output [17, 18].

Usually, the BPNN involves three or more layers. They are: input layer, single or multiple hidden layers, and output layer. The hidden and output layers contain separate activation functions; each layer performs its own activation function to produce an output signal [19].

The BPNN also involves an optimization algorithm for optimizing the output result. This algorithm is called the training function, which is responsible for training the BPNN network [20]. The BPNN network contains a library of training and activation functions, each of them is used for a specific task. The single training step is called epoch, or iteration.

The output that is presented by the output layer is determined by the following Eq. (1) [21]:

$$A=f_p(net_p) \tag{1}$$

where,

f_p: the activation of the output layer.

p: no. neurons of the output layer.

The output of the hidden layer can be determined by the following formula [21]:

$$B=f_q(net_q) \tag{2}$$

where,

f_q: activation function of the hidden layer.

q: No. neurons of the hidden layer.

The generated error function for updating the weights of the output layer can be formulated by the following equation [21]:

$$\delta_A=(d_p-A_p)\, f'(net_q) \tag{3}$$

where,

d: target of the output value of the output layer.

$f'(net)$: differentiation of $f(net)$.

The new updated weight that must be added to the previous connections' weights between hidden and output layers is estimated by the following formula [21]:

$$\Delta W=\eta\delta_A B' \tag{4}$$

where,

η: constant.

B': differentiation of B.

The generated error value which is used to update the weights connections between input and hidden layers can be determined by the following formula [21]:

$$\Delta B=w_j^t\delta_A f_B' \tag{5}$$

where,

w_j^t: tangential jth connection weight between input and output layer.

f_B': differentiation of activation function of the hidden layer.

The new weights which must be added to the previous weights between input and hidden layers can be determined by the following equation [21]:

$$\Delta V=\eta\ \delta_B\, q \tag{6}$$

where,

q: connection weights between input and hidden layers.

The output error signal of the output layer is represented by Mean Square Error (MSE), which can be formulated as follows [22]:

$$MSE=\sum_n(d-A)^2 \tag{7}$$

where,

n: number of output layer neurons.

d: target output vector.

A: actual output vector.

One of the important factors of the BPNN is the gradient function, which is defined as the differentiation of MSE with respect to weight [22]:

$$g=\delta(MSE)/\delta w \tag{8}$$

where,

w: weight of connections of the BPNN.

3.2 Proposed method

The BPNN was used as an intelligent system for a proposed controller of the clothes washing machine. This BPNN has been designed as a simulation form according to a prototype

of a simple washing machine which has the following specifications of input keys:

(1) Three steps of water levels: empty, 50%, and 100% Full.
(2) Two steps of washing levels.
(3) Two steps of rinsing levels.
(4) Two steps of drying (spin) levels.

After considering these specifications, the input of the proposed BPNN has been designed to involve ten input lines, and the output has been designed to involve five output lines. The proposed relation between input and output of the BPNN is illustrated in Table 1. The data set of this table represents one of 24 proposed modes of washing process. Each of these modes involves 8 steps. They differ only in the data set:

A0: Water level
A1: Wash level
A2: Rinse level
A3: Spin level
A4, A5: Water levels sensing
A6: Washing finish time
A7: Rinsing finish time
A8: Spinning finish time
A9: Start of process
B0: 2-way main motor working
B1: 1-way main motor working
B2: Water draining motor working
B3: Open/close the water entry
B4: Motor working time

The proposed BPNN network has been designed using the MATLAB software package by consideration of the following setting:

(1) The proposed BPNN network involves three layers: input, single hidden, and output layers.
(2) Ten neurons are included in the input layer, ten neurons for the hidden layer, and five neurons for the output layer. Usually, for each BPNN, one can start to choose two neurons in the hidden layer to get acceptable MSE results, but if we want to get an excellent result, we must increase the number of neurons step by step until the MSE reaches an excellent value. As known, increasing the number of neurons in the hidden layer leads to reducing the MSE, while increasing the number of hidden layers leads to increasing the MSE.
(3) All the weights of connections of the proposed network are set to a zero (0) value at the initial state.
(4) The initial Learning Rate (lr) is set to a zero (0) value, where it will be exponentially increased in the training process.
(5) The maximum of epochs is set to 220.
(6) The MATLAB library functions satlin and satlin have been used as activation functions for the hidden and output layers, respectively. They are shown in Figure 3. Satlin is a bipolar saturated linear function, while satlin is a unipolar saturated linear function. Specifically, these functions have been used because: (a) they are adequate with binary form input data to present excellent results, (b) the size of the proposed software can be minimized as much as possible.
(7) The MATLAB library function Traingda (means training by gradient descent algorithm with adaptive learning rate) has been used as a training function. This function is considered as an optimization method (algorithm) for the proposed network. It has been chosen because it can produce optimal results when used together with the selected linear activation functions satlin and satlin.

The proposed BPNN has been trained using the input and desired output data of Table 1. This table presents the eight steps of a simple washing process. These steps have been trained by the proposed BPNN.

After executing the MATLAB code program of the proposed BPNN, a simulation system block has been generated, which is shown in Figure 4. The system block is represented by the Custom Neural Network at the middle of the network. As shown in Figure 4, ten input ports (A0-A9) have been connected to the input layer of the system block through the Data type Converter. This converter converts the binary type of the input data to the decimal data of the system block. Also, five output (B0-B4) ports have been connected to the output layer of the system block through the Data Type Converter (1), where this converter converts the decimal data of the system block to binary form that is produced at the output ports. In this case, the signals of the programmed input switches and sensors are produced to the input ports (A0-A9), while the output ports present the output decisions to the motors and solenoids through the output ports (B0-B9).

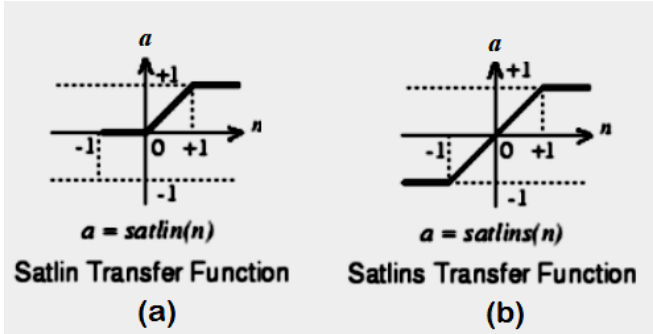

Figure 3. Characteristics of satlins and satlin activation functions [17]

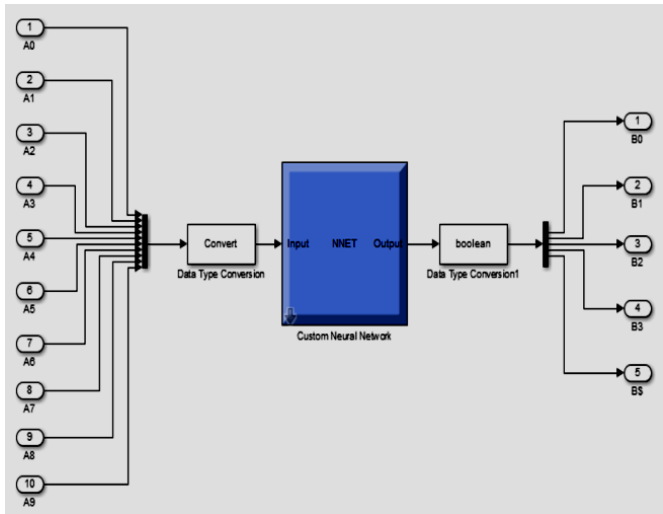

Figure 4. Block diagram of simulation of the proposed backpropagation neural network (BPNN) controller

As mentioned above, Table 1 illustrates the eight required steps for one of the washing programs of clothes washing processes. The first step of this program begins with the input code (0000000001), which represents the starting mode. In this state, the program begins with the first level (0) for the water, wash, rinse, spin, and water level sensing. Also in this state, the washing, rinsing, and spinning finish time at the first level. A9 is in a (High) state, which refers to the beginning of

the washing process. When this code is entered into the input of the proposed BPNN network, an output code (00010) is generated at the output of the proposed network. This output code refers to activating the solenoid of the pure water entry, which allows the pure water to enter the tube of the washing machine. The second step, when the tube is half full, the code (0000010001) is introduced to the input of the proposed network, so the network will present the output code (10000) which refers to starting the working of the main motor with two-way rotation to wash the clothes. The third step, when the washing time is finished, the input code (0000011001) will be presented at the input of the proposed network, so, the network will present the output code (00100) that refers to start the draining of the washing water by the water pump. The fourth step, the input code (0000001001), is entered into the input of the proposed BPNN, which refers to the finishing of emptying the tube by the draining water pump, so the output code (00010) will be produced at the output of the proposed BPNN. This output code refers to turn off the draining water pump and turn on the solenoid of the water entry gate to allow the water to enter to the tube again.

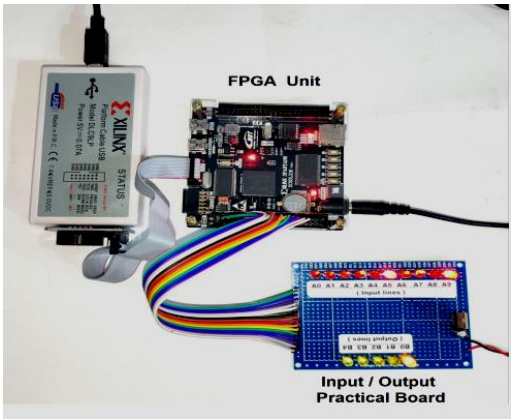

Figure 5. Practical implementation of the proposed backpropagation neural network (BPNN) controller in operation mode

The next steps for rinsing and then drying the clothes are performed in the same manner until the clothes are dried while the washing process is finished. After realizing the proposed BPNN controller, it has been converted to a VHDL code, then this VHDL program has been implemented in FPGA. The FPGA which has been used in this work is SPARTAN6-XC6SLX16, it is simple, low cost, and fast, it has clock frequency 50MHz, the maximum software size is 16 Mbyte, and the cost of this FPGA is nearly $35. Figure 5 illustrates the complete wire connection between the experimental board and the FPGA in operation mode, as shown: the Light Emitting Diodes LEDs of A5 and A9 are in ON state, which represents the second step of the Table 1, and the LED of B0 in ON state, this operation mode represents the state of operating the AC motor the washing mode, and the tube of the washing machine is half full.

4. RESULTS AND DISCUSSION

After training the proposed BPNN using the data set in Table 1, a report has been presented on the screen of the computer. It includes the highlight results of testing the network. This report is shown in Figure 6, as illustrated. This figure includes:(1) the block diagram of the proposed network, (2) the algorithm that has been used in the network, and (3) the progress results that include the values of no. epochs, performance, and gradient. These results are presented in the plot field. They are: Performance, training state, and regression. By clicking on the performance soft key, the performance plot will be presented. This plot represents the relation between the MSE and no. iterations (epochs) of the complete training process. The performance plot is illustrated in Figure 7. As shown in this plot, one can see the MSE starts from a value of nearly 0.05 at epoch 0, and then gradually reduces to zero value at epoch 90, which is considered an exciting result. This result supports and confirms the reliability of the proposed BPNN.

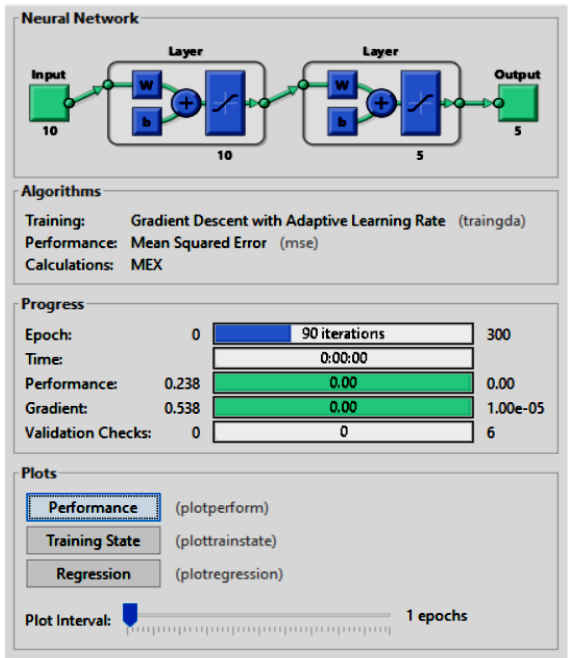

Figure 6. Report of testing the proposed backpropagation neural network (BPNN) controller

Figure 7. Performance plot of the proposed backpropagation neural network (BPNN) controller

By training the data sets of the remaining 23 modes, they all present the same MSE results.

By clicking on the Training State soft, the training state plot will be present on the screen of the computer, which is shown in Figure 8. This figure includes three important plots: (1)

Gradient plot, (2) Validation plot, and (3) Learning rate plot. The Gradient plot illustrates that the gradient gradually reduces from 0.5 at epoch 0 to 105 at epoch 90, which enhances the brilliant success of the training process. The validation plot shows that the validation check of the used data is zero (0) at all epochs, i.e. all the used data are valid during the training process. The learning rate plot illustrates that the learning rate increases exponentially from a zero (0) value at epoch 0 to nearly 4 at epoch 90 according to the choice of adaptive learning rate due to using the activation function Traingda in the training process.

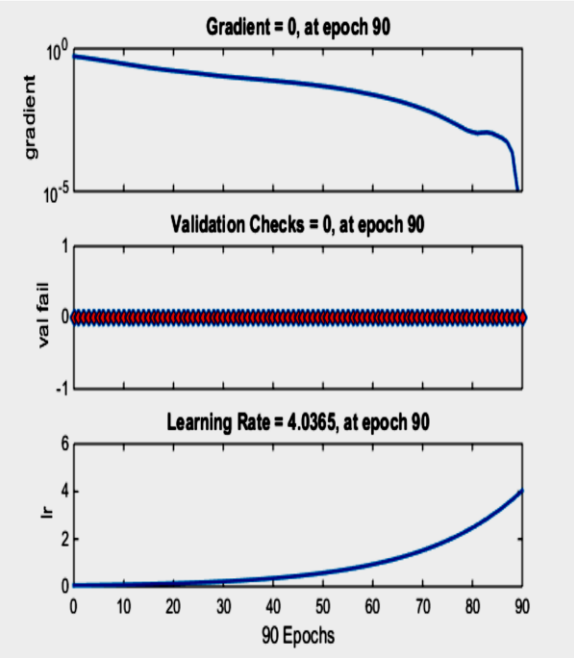

Figure 8. Training state plots of the proposed backpropagation neural network (BPNN) network

By clicking on the Regression soft key, the regression plot will be presented, which is shown in Figure 9. This plot represents the relation between the real and target outputs. As seen from this plot, a solid blue line represents the fitting between the real and target outputs during the training process. This fitting confirms the high reliability of operation of the proposed BPNN controller.

The processing time of the proposed BPNN has been obtained using the MATLAB package tools. It depends on the simplicity of the proposed software, and it also depends on the processing time of the used personal computer that performs the program. In this work, a fast computer was used to perform the proposed software. It has a core i9 processor type with a 4 GHz clock frequency, and the obtained processing time is nearly reached to (2-3) ns. This value is much less than that of microcontrollers that are used in modern washing machines, such as the "Mitsubishi FX3U-64MT/ESS", which has a maximum processing time of 12.5 µsec. The processing time of the proposed controller after implementing in FPGA is increased to nearly 60 ns due to the limited processing speed for one machine cycle is 20 ns of the used FPGA (type: SPARTAN6- XC6SLX16) that has clock frequency 50MHz. This FPGA has been used due to the following characteristics: low cost, simple, and available in markets. If a faster FPGA that has a clock frequency of 500 MHz is used, one can get less than 60 nsec processing time. The consumption power of this FPGA is in the range (23–33) mW, which is lower than that of microcontrollers.

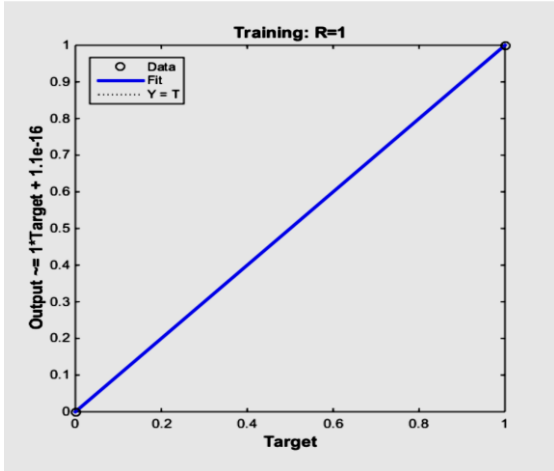

Figure 9. Regression plot of the proposed backpropagation neural network (BPNN) controller

A comparison has been carried out between the results of the proposed and related works that is included in Table 2. As shown, the proposed work uses different training functions. This function depends on a technique of adaptive increasing the learning rate, which can help to present excellent results. The proof of that, the value of the obtained MSE has reached zero. Referring to this table, one can see the significant difference between the MSE value of the proposed work of this manuscript and that of the related works. Another important difference, all the related work is not implemented, they are just simulations or software, while the proposed work has been implemented in an FPGA.

Another report has been presented after converting the proposed MATLAB simulation to VHDL code. This report illustrates the details of the number of used items in the FPGA, which is shown in Figure 10.

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	567	93,120	1%
Number used as Flip Flops	566		
Number used as Latches	1		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	769	46,560	1%
Number used as logic	739	46,560	1%
Number using O6 output only	489		
Number using O5 output only	15		
Number using O5 and O6	235		
Number used as ROM	0		
Number used as Memory	0	16,720	0%
Number used exclusively as route-thrus	30		
Number with same-slice register load	28		
Number with same-slice carry load	2		
Number with other load	0		
Number of occupied Slices	304	11,640	2%
Number of LUT Flip Flop pairs used	905		
Number with an unused Flip Flop	415	905	45%
Number with an unused LUT	136	905	15%
Number of fully used LUT-FF pairs	354	905	39%
Number of unique control sets	51		
Number of slice register sites lost to control set restrictions	209	93,120	1%

Figure 10. Report of downloading the proposed VHDL code into FPGA

Note: FPGA = Field-Programmable Gate Array; VHDL = Very High Speed Integrated Circuit Hardware Description Language

Table 2. Comparison between results of the proposed and related works

Work	Network Layers	Training Function	Learning Rate	No. Epochs	Mean Square Error (MSE)	Implemented	Software Size
[14]	17-5-10	Trainpso	0.0001	50	0.11	NO	4.7 MByte
[15]	20-10-10	Trainlm	0.05	300	0.045	NO	5.6 M MByte
[16]	3-9-3	$G(s) = \frac{e^{-90s}}{280s+1}$	0.3	200	0.038	NO	4.1 MByte
The proposed work in this manuscript	10-10-5	Traingda	Adaptive increasing from 0 to 4	90	zero	Yes Implemented into FPGA	3.1 MByte

As shown, the number of used slices LUTs (Look up Tables) is 769, while the remaining is 46560, and the number of used flip-flops is 566, and the number of slice registers is 567, while the remaining is 93120.

According to these results, one can conclude that the proposed BPNN software has occupied a small part of the FPGA, where no available items of most components constitute a small percentage of the full capacity of the FPGA, which also refers to success the achievement success of the proposed work.

5. CONCLUSIONS

The proposed software of this work presents stimulating results using an eight-step data set of the washing process. If the steps are increased much more than this number, then the MSE can be increased. Also, if the number of data bits is increased, the MSE value can be increased too. Using the linear activation functions supports the simple conversion from MATLAB software to VHDL code, and at the same time reduces the software size. If non-linear functions are used, it can present better results, but the proposed software size and the complexity of the software can be increased, which is not preferred. Using a binary type of data set, the linear activation functions can present better results than using non-linear functions. BPNN networks can be used for all electronic control applications, and they can also be easily implemented in microprocessor-based systems or FPGA. The size of the BPNN software forces us to choose the appropriate FPGA to implement this software. The proposed application has used the Traingda training function and presents very accepted results. For future work, other training functions can be used for presenting better results, such as: Trainlm (Leven-berg training), Traingd (gradient descent training). The learning rate is a very important parameter for training the BPNN network. Therefore, the adaptive increase of this parameter leads to a reduction in the no. iterations (epochs), which in turn leads to speed up the training process.

Reducing the size of the BPNN network supports the speed-up of the processing, and makes the designer to choose a simpler and lower-cost FPGA, whereas, the cost of FPGA depends on:(1) processing speed (or clock frequency), (2) number of internal components, Programmable Logic Blocks PLBs, Input/Output blocks IOBs, and no. processing data bits.

REFERENCES

[1] Sembiring, E. (2024). The effect of top-loading washing operational setting on microplastic fibers released from cloth during the washing process and filtered by filter cloth. Asian Journal of Engineering, Social and Health, 3(1): 34-46. https://doi.org/10.46799/ajesh.v3i1.199

[2] Olalekan Salau, A., Takele, H. (2022). Towards the optimal performance of washing machines using fuzzy logic. Scientific Programming, 2022(1): 8061063. https://doi.org/10.1155/2022/8061063

[3] Yang, B. (2023). A vector equation method for analyzing kinematics and kinetostatics of toggle-type transmission mechanism. Journal of Intelligent Systems and Control, 2(1): 23-32. https://doi.org/10.56578/jisc020103

[4] Mishra, S. (2023). An innovative approach in modelling and design of smart washing machine with automatic drying with estimating energy and water consumption using AI. In 2023 International Conference on Power Energy, Environment & Intelligent Control (PEEIC), Greater Noida, India, pp. 175-180. https://doi.org/10.1109/PEEIC59336.2023.10450535

[5] Erdle, L.M., Nouri Parto, D., Sweetnam, D., Rochman, C.M. (2021). Washing machine filters reduce microfiber emissions: Evidence from a community-scale pilot in Parry Sound, Ontario. Frontiers in Marine Science, 8: 777865. https://doi.org/10.3389/fmars.2021.777865

[6] Jasiūnas, Ž., Julião, T., Cecílio, J., da Graca, G.C., Ferreira, P.M. (2025). A soft sensor to assess the energy performance of laundry washing machines. Applied Energy, 383: 125349. https://doi.org/10.1016/j.apenergy.2025.125349

[7] Puentes, K., Morales, L., Pozo-Espin, D.F., Moya, V. (2024). Enhancing control systems with neural network-based intelligent controllers. Emerging Science Journal, 8(4): 1243-1261.

[8] Gao, Y., Zhu, G., Zhao, T. (2022). Based on backpropagation neural network and adaptive linear active disturbance rejection control for attitude of a quadrotor carrying a load. Applied Sciences, 12(24): 12698. https://doi.org/10.3390/app122412698

[9] Liu, H., Yu, Q., Wu, Q. (2023). PID control model based on back propagation neural network optimized by adversarial learning-based grey wolf optimization. Applied Sciences, 13(8): 4767. https://doi.org/10.3390/app13084767

[10] Saeed, A.B., Gitaffa, S.A., Dawai, R.I. (2025). FPGA-based realization of intelligent escalator controller using artificial neural network. Journal of Electrical and Computer Engineering, 2025(1): 7567924. https://doi.org/10.1155/jece/7567924

[11] Zhu, Y., Hou, K. (2024). Development and implementation of an FPGA-embedded multimedia remote monitoring system for information technology server room management. International Journal of Digital Multimedia Broadcasting, 2024(1): 4420578. https://doi.org/10.1155/2024/4420578

[12] Li, H., Wan, B., Fang, Y., Li, Q., Liu, J.K., An, L. (2024). An FPGA implementation of Bayesian inference with spiking neural networks. Frontiers in Neuroscience, 17: 1291051. https://doi.org/10.3389/fnins.2023.1291051
[13] Tran-Thi, B.N., Nguyen-Ly, T.T., Hoang, T. (2023). An FPGA design with high memory efficiency and decoding performance for 5G LDPC decoder. Electronics, 12(17): 3667. https://doi.org/10.3390/electronics12173667
[14] Yan, Y., Chen, R., Yang, Z., Ma, Y., et al. (2022). Application of back propagation neural network model optimized by particle swarm algorithm in predicting the risk of hypertension. The Journal of Clinical Hypertension, 24(12): 1606-1617. https://doi.org/10.1111/jch.14597
[15] Mohammed, N.A., Al-Bazi, A. (2022). An adaptive backpropagation algorithm for long-term electricity load forecasting. Neural Computing and Applications, 34(1): 477-491. https://doi.org/10.1007/s00521-021-06384-x
[16] Zhang, X. (2025). Design and application of PLC control system based on neural network. Procedia Computer Science, 261: 1005-1011. https://doi.org/10.1016/j.procs.2025.04.493
[17] Gitaffa, S.A., Issa, A.H., Ibrahem, Y.N. (2022). Deep neural network technique based Field Digitizing Units (FDUs) instruments fault detection and isolation. Journal Européen des Systèmes Automatisés, 55(5): 657-663. https://doi.org/10.18280/jesa.550511
[18] Pei, J.F., Zhang, J., Jin, D.C., Miao, B.L. (2024). Backpropagation neural network-based survival analysis for breast cancer patients. International Journal of Radiation Research, 22(1): 163-169. https://doi.org/10.52547/ijrr.21.23
[19] Saeed, A.B. (2021). Elevator controller based on implementing a random access memory in FPGA. International Journal of Electrical and Computer Engineering, 11(2): 1053-1062. https://doi.org/10.11591/ijece.v11i2.pp1053-1062
[20] Hashem, I.A.T., Alaba, F.A., Jumare, M.H., Ibrahim, A.O., Abulfaraj, A.W. (2024). Adaptive stochastic conjugate gradient optimization for backpropagation neural networks. IEEE Access, 12: 33757-33768. https://doi.org/10.1109/access.2024.3370859
[21] Dam, Q.T. (2024). Adaptive neural network-based PID controller design for velocity control of an internal combustion engine using back propagation technique. Journal of Electronics and Electrical Engineering, 3(2): 613-630. https://doi.org/10.37256/jeee.3220245581
[22] Zhu, X., Li, M., Liu, X., Zhang, Y. (2024). A backpropagation neural network-based hybrid energy recognition and management system. Energy, 297: 131264. https://doi.org/10.1016/j.energy.2024.131264

NOMENCLATURE

n	number of neurons in output layer
d	desired output vector
o	real output vector
w	connections weight of the BPNN
g	gradient function
f_q	activation of the output layer
q	no. neurons of the output layer
d	target of output value of the output layer
$f'(net)$	differentiation of $f(net)$
η	constant
B'	differentiation of B
w_j^t	tangential j^{th} connection weight between input and output layer
q	connection weights between input and hidden layers