energies

MDPI

Article

Condition Assessment of Power Transformers through DGA Measurements Evaluation Using Adaptive Algorithms and Deep Learning

Dimitris A. Barkas, Stavros D. Kaminaris, Konstantinos K. Kalkanis, George Ch. Ioannidis and Constantinos S. Psomopoulos *

Department of Electrical and Electronics Engineering, University of West Attica, GR-12244 Egaleo, Greece
* Correspondence: cpsomop@uniwa.gr

Abstract: Condition assessment for critical infrastructure is a key factor for the wellbeing of the modern human. Especially for the electricity network, specific components such as oil-immersed power transformers need to be monitored for their operating condition. Classic approaches for the condition assessment of oil-immersed power transformers have been proposed in the past, such as the dissolved gases analysis and their respective concentration measurements for insulating oils. However, these approaches cannot always correctly (and in many cases not at all) classify the problems in power transformers. In the last two decades, novel approaches are implemented so as to address this problem, including artificial intelligence with neural networks being one form of algorithm. This paper focuses on the implementation of an adaptive number of layers and neural networks, aiming to increase the accuracy of the operating condition of oil-immersed power transformers. This paper also compares the use of various activation functions and different transfer functions other than the neural network implemented. The comparison incorporates the accuracy and total structure size of the neural network.

Keywords: dissolved gas analysis; neural networks; adaptive algorithm

Citation: Barkas, D.A.; Kaminaris, S.D.; Kalkanis, K.K.; Ioannidis, G.C.; Psomopoulos, C.S. Condition Assessment of Power Transformers through DGA Measurements Evaluation Using Adaptive Algorithms and Deep Learning. *Energies* **2023**, *16*, 54. https://doi.org/10.3390/en16010054

Academic Editor: Abu-Siada Ahmed

Received: 6 November 2022
Revised: 11 December 2022
Accepted: 15 December 2022
Published: 21 December 2022

1. Introduction

The term power quality of electricity is one of the most discussed in the last decades. Modern lifestyle, developed through technological evolution, leads to an increased demand for electrical power. Additionally, the continuous development and installation of renewable sources in combination with distributed power generation and the high penetration of electrical vehicles and power electronics have led to the change of the load behavior from resistive to capacitive. Additionally, the effective transmission of electricity plays a key role for the future power grid and its stability. New approaches to this field have been proposed [1]. However, existing electricity networks have been designed many years ago, obligating the equipment to work to the limits, and to operate under capacitive loads (exceeding initial design specifications). These two key factors are usually responsible for the malfunction and failure of critical power equipment such as power transformers. The majority of power transformers are of the oil-immersed type mainly due to their high utilization, in the sense that they offer high electric power management (voltage step-up or step-down) with small size in comparison to other power transformer types such as the dry type. The main reason that a power transformer malfunction is the deterioration of its insulation. Power transformers are devices that transport and distribute the electrical energy point to point by voltage step-up or step-down. The transportation is achieved through the high voltage and ultra-high voltage electrical transportation system with typical voltage levels higher than 150 kV. At these voltage levels, the insulation is stressed due to the high electrical field, the deterioration rate being dependent on the current insulation and environmental conditions (usually temperature, humidity, surface dust, and UV radiation).

Knowledge of the insulation conditions can render significant results regarding the lifetime of a power transformer, as it is closely related to the insulation lifetime.

The condition assessment of the insulation is of great importance for the high and medium power transformers. As already mentioned, these transformers are of the oil-immersed type making these devices extremely complicated of the physicochemical reactions that take place inside them, affecting the insulation's condition. The insulation is comprised of solid insulation parts such as the bushings, the windings' external insulation, the craft paper, liquid insulation, and the insulating oil. Many key parameters can be measured concerning the insulating parts of power transformers, with the most common being the DGA (Dissolved Gas Analysis) of the insulating oil, the tangent delta (tan d) or power factor, partial discharges, and insulation resistance [2,3]. DGA is the most widely applied technique in modern monitoring systems, and the application can be achieved either online or offline. This technique is based on the analysis of the insulating oil for the existence of specific key gases. These gases are hydrogen (H_2), oxygen (O_2), nitrogen (N_2), carbon monoxide (CO), carbon dioxide (CO_2), methane (CH_4), ethane (C_2H_6), ethylene (C_2H_4), acetylene (C_2H_2), propane (C_3H_8), and propylene (C_3H_6) [4]. The existence of different key gases in the oil and varying concentrations reveals different kinds of insulation-related problems. These key gases are usually utilized in the form of ratios, as several DGA techniques propose. The most known DGA analysis techniques are the IEC ratio method, Doerenburg ratio method, Rogers' ratio method, and Duval's triangle method. The IEC and Roger ratio methods use ratios C_2H_2/C_2H_4, CH_4/H_2, C_2H_4/C_2H_6, whereas the Doerenburg ratio method uses ratios CH_4/H_2, C_2H_2/C_2H_4, C_2H_2/CH_4, C_2H_6/C_2H_4, and the Duval's triangle uses the gases CH_4, C_2H_4, and C_2H_2. DGA analysis techniques are probably the most important tools for the condition assessment of power transformers. However, they should not be applied independently but as a supportive tool to the condition assessment [5–12].

From past experience [13–15] when the application of DGA methods has not proved conclusive, the correct diagnosis of an insulation problem has therefore failed. For this reason, new techniques in the field of condition assessment have been applied in the last two decades. These techniques emanate from the novel and vast scientific area of Artificial Intelligence (AI) [6,7,9,12,16]. The use of AI techniques is not limited to the DGA for the condition assessment of electricity networks: they can also be applied in other scientific areas of the electricity network including the economic operation of power grids using load frequency control [17] and profit maximization of renewable power producers [18].

The combination of machine learning with DGA analysis techniques leads to better results compared to the solution without machine learning. Such combinations include the Duval triangle; ANN without hidden layers [6]; IEC ratio and Rogers ratio methods combined with ANN [7]; back-propagation ANN (BPNN); extreme learning machine–radial basis function (ELM–RBF); fast computed Neural Networks; Support Vector Machines (SVM) [12]; and Fuzzy Logic [9,16]. However, most of the research [6,7,9,12,16] illustrates simulations for standard artificial intelligence techniques configurations and structures with no explanation or verification of the structure selection. Such an example is the selection of the number of hidden layers and the neurons' inputs and outputs. Accuracies presented in the said simulations reach up to 88.04% in training data for BPNN with slow computation time, and 86.12% in training data for SVM with fast computation time. The solution presented in this article achieves an accuracy of 86.6% with the training data but with the neural network's structure parameters calculated automatically without extra human research effort. This manuscript proposes a first approach to the construction of an algorithm which selects an optimized value for many parameters like the hidden layers' number value and the number of neurons for each layer. This algorithm, which is presented in Section 3 in detail, describes the application of such a technique called Multilayer Neural Network (MLNN). This technique requires a set of input parameters to export a specific result. Its work operation is described in the following section. The proposed MLNN receives the values of the specific gases described above as inputs while it is able of problem

recognition that may exist in the oil–immersed power transformer. The possible problems can be:

- ➢ Partial Discharge
- ➢ Spark Discharge
- ➢ Arc Discharge
- ➢ High—Temperature Overheating
- ➢ Middle—Temperature Overheating
- ➢ Low—Temperature Overheating
- ➢ Low/Middle Temperature Overheating

The data and the above-mentioned states were collected by the IEEE DGA datasets [19]. The use of these datasets requires the knowledge that the transformer may be in a fault state and the MLNN proposed can be used as an auxiliary diagnosis tool.

This paper is structured into five sections. Section 1 being the Introduction, the current state of the DGA in power transformers and Artificial Intelligence is presented. In Section 2 there is a reference to machine learning and artificial intelligence, and a more detailed description of the MLNNs. Section 3 describes the adaptive algorithm implemented for the NN structure selection. In Section 4, the results of the applied algorithm are presented followed by comparison, while Section 5 describes the resume of the manuscript and future optimization proposals.

2. Artificial Intelligence

The engineering community has made a significant shift in the last two decades toward the designing and building of machines that can think and act like a human brain. In other words, programming machines to develop intelligence is currently a popular trend. The scientific area that examines this effort is called "Artificial Intelligence" (AI) and can be considered as a computer science branch which is divided into many subareas due to the recent surge in research on this field. Figure 1 illustrates the sub-branches of AI [20–24]. The basic concept of AI is the description and the design of agents that receive stimuli from the environment and perform actions. The basic subareas of AI are perception, machine learning, robotics, natural language processing, reasoning, decision making, and expert systems. Many readers and scientists usually confuse the term machine learning with the term AI; however, as is clear from Figure 1, machine learning (ML) is a kind of AI [20].

In this paper, the application of a specific method of AI is used to categorize the status of oil-immersed power transformers. This method is named Neural Networks (NN) and, as indicated in Figure 2, this method belongs to the ML subarea [20,22,24]. Machine learning is divided into two subcategories, the first one includes supervised learning methods, while the other includes unsupervised learning methods. Both methods aim to render an answer to a specific problem such as the one regarding the power transformer's operating conditions. By using these techniques, any problem can be solved, but some kind of input data is necessary. The idea is to build a prediction system that, through a process called "Training", maps the inputs to particular outputs, much like a human is trained to perform tasks. The system is started out in this training process with no knowledge of the issue or the solution. This system is fed with input and corresponding output data and its internal structure is modified until it achieves a satisfactory percentage of success. In the case of the problem of the operating condition of oil-immersed power transformers, the input data can be the gases produced inside the transformer due to the overloading, and the output data could be the condition characterization such as "overheating". In the case of supervised learning algorithms, they develop predictive models based on input and output data, which means that these algorithms use specific training sets to train the system whereas unsupervised learning methods group and interpret the data, only based on the input data, which means that no training sets are required [21].

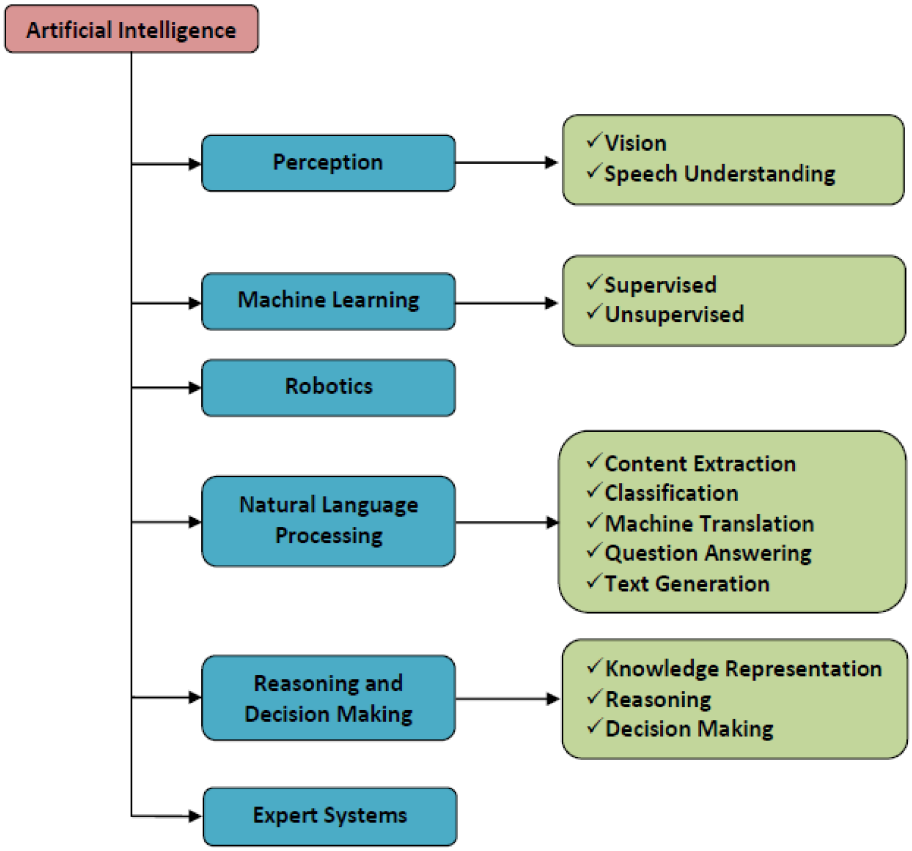

Figure 1. Artificial intelligence categorization [20,22–24].

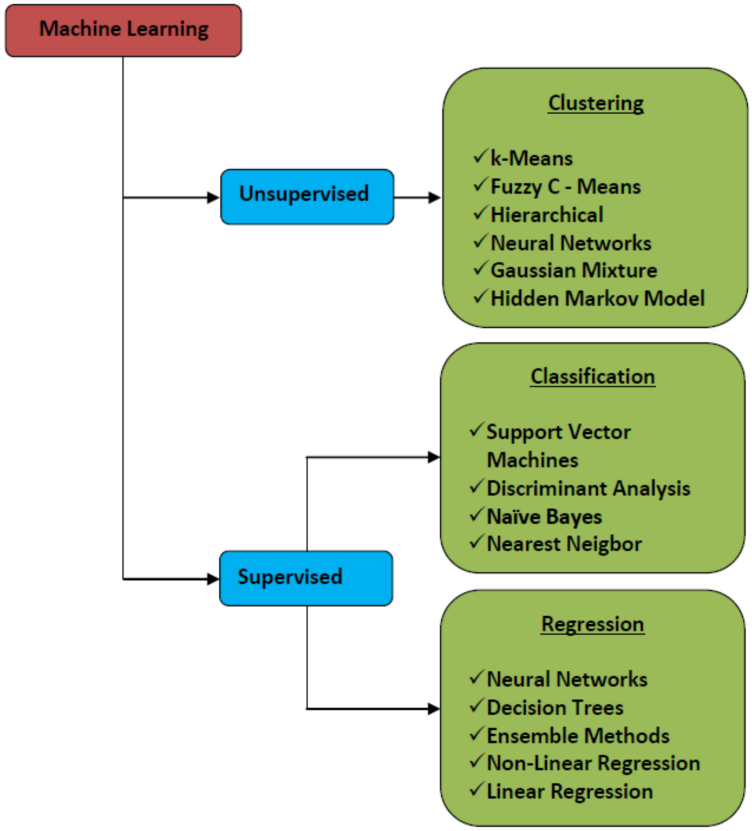

Figure 2. Machine learning categorization [20,22,24].

As the NN is the method used in this paper, a swift introduction of it is of great importance to its main description and the main parameters that affect it. Neural networks are processing structures that consist of simple processing units working in parallel. These structures have the advantage of storing experimental knowledge for future use. This knowledge is retrieved by the environment through a learning process, while the connection between the different simple processing units is dependent on factors called "weights", and they are used for storing acquired knowledge. There are many reasons for the selection of NN as the method of solution of a problem such as the nonlinearity, the input-output mapping, the adaptivity, the evidential response, the fault tolerance, and the VLSI implementability. Nonlinearity arises due to the construction of NNs and the interconnections of the simple processing elements. The input-output mapping is the property of the NN to learn how to calculate the output using the inputs and altering the weights' values through a learning processing (supervised learning). Adaptivity arises through the capacity to change the weights of the interconnection of NN structure because of the surrounding environment. The evidential response, in the case of pattern classification, is the ability not only to select a specific pattern but also the level of confidence in the decision made. Additionally, the implementation of NNs on hardware leads to fault-tolerant structures, while the parallel connected nature of simple elements makes NNs capable to implement in hardware modules such as VLSI (Very Large Scale Integrated) chips [25].

A simple processing element constructing NNs, called neuron, is presented in Figure 3. A neuron is structured by three basic elements:

- **Connecting links**: Connect the inputs of the neuron to the adder (next element of the structure) through a weight.
- **Adder**: Sums all the values come from the connecting links
- **Activation function**: Applies a function of one variable, with this variable's value being the result of the adder unit.

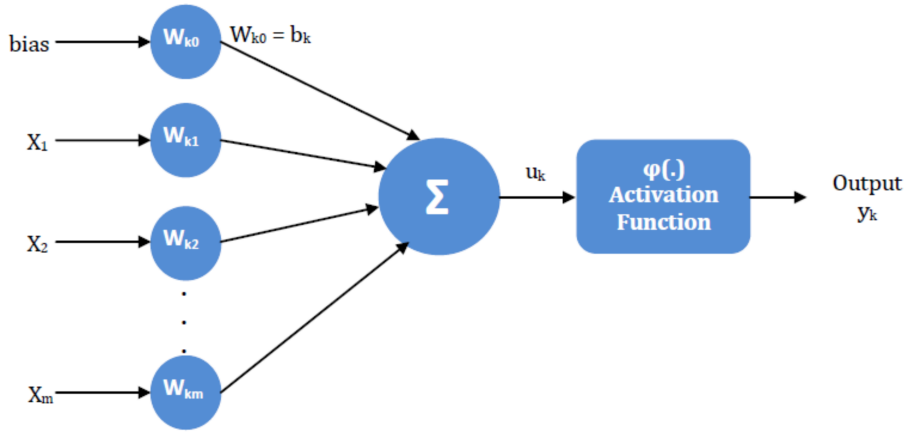

Figure 3. The basic structure element of neural networks, called neuron.

In Figure 3 the neuron has $m + 1$ inputs, x_1 to x_m, the b_k called bias, and an output y_k. In some cases, the bias is referred to as input without a weight, but its manipulation as an input with a weight helps the mathematical analysis. The output range is in the range of $[0, 1]$ or $[-1, 1]$. The x is referred to as the input variables of the problem while the bias increases or lowers the net input of the activation function, depending on whether it is positive or negative, respectively. The output of the adder is a signal u_k while the output y_k is the result of the activation function when the value of its variable is u_k. In mathematical terms it can be written as:

$$u_k = \sum_{j=1}^{m} \left\{ w_{kj} \cdot x_j \right\} \tag{1}$$

$$y_k = \varphi(v_k) \tag{2}$$

Many activation functions can be used with the most known being the step or threshold function and the sigmoid function with varying slope parameter. Tables 1 and 2 present possible activation functions and loss functions respectively, that could be used in NN according to the literature [26,27].

Table 1. Most known activation functions in NNs [26,27].

Function Name	Function
Threshold function	$f(x) = \begin{cases} 1\ , & x \geq 0 \\ 0\ , & x < 0 \end{cases}$
Sigmoid function	$f(x) = \frac{1}{1+e^{-x}}\ ,\ x \in (-\infty, +\infty)$
Tanh	$f(x) = 2 \cdot sigmoid(2x) - 1 = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
(Rectified Linear Unit) ReLU	$f(x) = \begin{cases} 0\ , & x_i < 0 \\ x\ , & x \geq 0 \end{cases}$
Leaky ReLU	$f(x) = \begin{cases} a \cdot x\ , & x < 0 \\ x\ , & x \geq 0 \end{cases}$ where a small real number (e.g., 0.01)
Parametric Relu	$f(x) = \begin{cases} a_c \cdot x\ , & x < 0 \\ x\ , & x \geq 0 \end{cases}$ where a real number
Exponential Linear Unit (ELU)	$f(x) = \begin{cases} a \cdot (e^x - 1),\ & x < 0 \\ x\ , & x \geq 0 \end{cases}$ where a is a real number
Piecewise Linear Deformable Exponential Linear Unit (PDELU)	$f(x) = \begin{cases} x\ , & x > 0 \\ a \cdot \left([1 + (1-t) \cdot x]^{\frac{1}{1-t}} - 1 \right), & x \leq 0 \end{cases}$ where a, t are real numbers. Parameter a controls the negative slope of function, and parameter t controls the degree of deformation
Swish	$f(x) = \frac{x}{1+e^{-\beta x}}$ where parameter "β" is a learnable parameter
Mish	$f(x) = x \cdot \tanh(ln(1 + e^{ax}))$ where parameter "a" is a learnable parameter
TanELU	$f(x) = ReLU(x) + a \cdot \tanh(x)$ where parameter "a" is a learnable parameter
S-shaped ReLU (SReLU)	$f(x) = \begin{cases} t^l + a^l(x - t^l),\ & x \leq t^l \\ x\ , & t^l < x < t^r \\ t^r + a^r(x - t^r),\ & x \geq t^r \end{cases}$ where a is a real number and t^l, t^r, a^l and a^r are four learnable parameters
Adaptive Piecewise Linear Unit (APLU)	$f(x) = ReLU(x) + \sum_{c=1}^{n} a_c \cdot \min(0, -x,\ b_c)$ where a_c and b_c are real numbers
Mexican ReLU (MeLU)	$f(x) = PReLU^{c_0}(x) + \sum_{j=1}^{k-1} c_j \cdot \varphi^{a_j, \lambda_j}(x)$ where k is the number of learnable parameters for each channel, c_j are the learnable parameters, and c_0 is the vector of parameters in PReLU
Gaussian ReLU (GaLU)	$f(x) = PReLU^{c_0}(x) + \sum_{j=1}^{k-1} c_j \cdot \varphi_g^{a_j, \lambda_j}(x)$ where k is the number of learnable parameters for each channel, c_j are the learnable parameters, and c_0 is he vector of parameters in PReLU
Soft Root Sign (SRS)	$f(x) = \frac{x}{\frac{x}{\alpha} + e^{-\frac{x}{\beta}}}$ where α, β are nonnegative learnable parameters
Soft Learnable	$f(x) = \begin{cases} x\ , & x > 0 \\ a \cdot ln\left(\frac{1+e^{\beta x}}{2}\right), & x \leq 0 \end{cases}$ where α and β are nonnegative trainable parameters
Splash	$f(x) = APLU_{a_i^+, b_i}(x) + APLU_{a_i^-, b_i}(-x)$ where α_i, β_i are learnable parameters
SoftMax	$f(x) = \frac{e^x}{\sum_{j=1}^{n} e^x}$ where n is the number of classes (possible outcomes)

Table 2. Loss functions for NN.

Function Name	Expression
Quantifying Loss	$\mathcal{L}(f(x^{(i)};W),\, y^{(i)})$
Empirical Loss	$J(W) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x^{(i)};W),\, y^{(i)})$
Binary Cross Entropy Loss	$J(W) = -\frac{1}{n}\sum_{i=1}^{n} \quad y^{(i)} log\left(f(x^{(i)};W)\right) + (1-y^{(i)}) \log\left(1-\left(f(x^{(i)};W)\right)\right)$
Mean Squared Error Loss	$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f(x^{(i)};W)\right)^2$

where $f(x^{(i)};W)$ is the predicted value and $y^{(i)}$ is the actual value.

The application of NN as a problem-solving approach requiring the decision of a specific structure for the NN. There are three basic structure categories for NN.

- ➢ Single–Layer Feedforward Networks
- ➢ Multilayer Feedforward Networks
- ➢ Recurrent Networks

In the Single–Layer Feedforward Networks there is a unique layer of neuron nodes that accepts the inputs and produces the outputs, as Figure 4 presents. The outputs of the layer do not feedback and for this reason, this structure is called also feedforward. The input layer (small rectangles) is not considered a layer because there is no computation performed there.

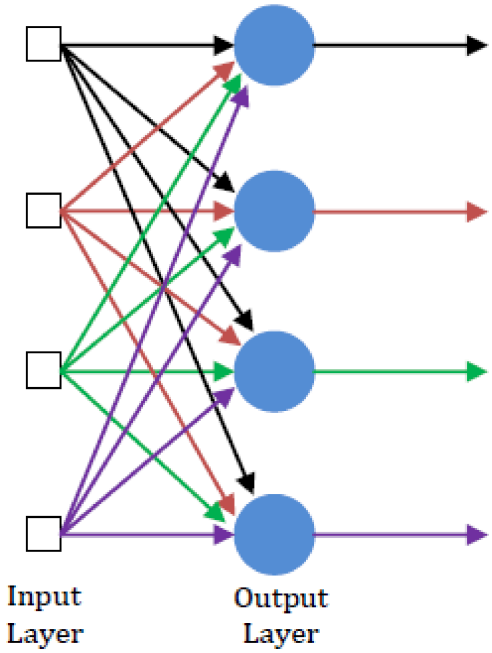

Figure 4. The single—layer feedforward network.

The multilayer feedforward network structure is an extension of the single-layer feedforward network. In this case, as Figure 5 presents, except for the input layer and the output layer, there is an intermediate layer of nodes which computes the output. The intermediate layer (the layer with the four nodes) is called the hidden layer and inside its nodes a series of internal NN computations take place. A multilayer feedforward NN can have many hidden layers, while their number is limited by the application and the machine which runs the NN. As in the case of single—layer feedforward structure, the computations

and the total information start from the input layer and conclude at the output layer with no feedback loops.

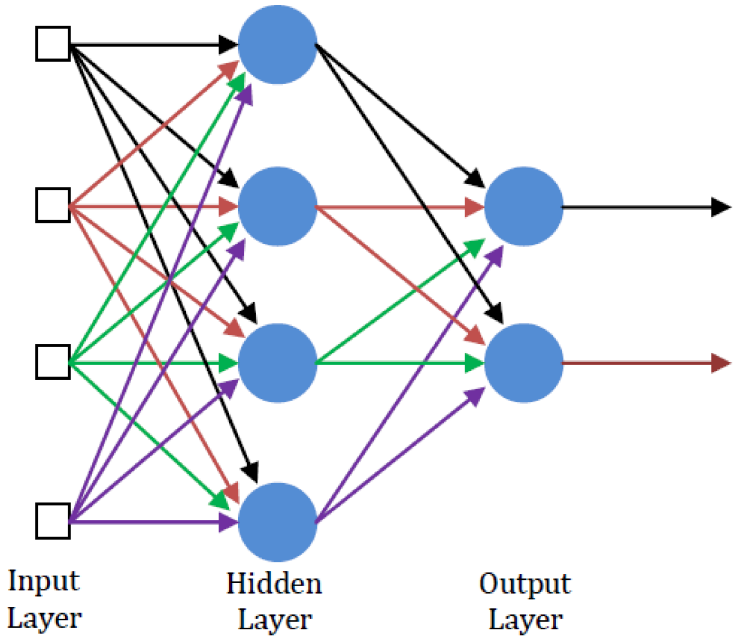

Figure 5. The multilayer feedforward network.

The last basic structure is the Recurrent Networks, and a simple structure is illustrated in Figure 6. The main difference between a recurrent network and a feedforward network is that in the recurrent networks there is at least one feedback loop. In the structure in Figure 6, there is a unique layer of neurons, no hidden layer, and each neuron feeds the inputs from the output signals. Furthermore, in this structure there is no feedback from its neuron to itself, which means there is no self-feedback loop. An extension of this structure is presented in Figure 7 where there are hidden neurons. The existence of feedback loops in recurrent neural networks affects the learning capability of the network and its performance. Additionally, there are unit-time delays in the feedback loops leading to nonlinear dynamic systems.

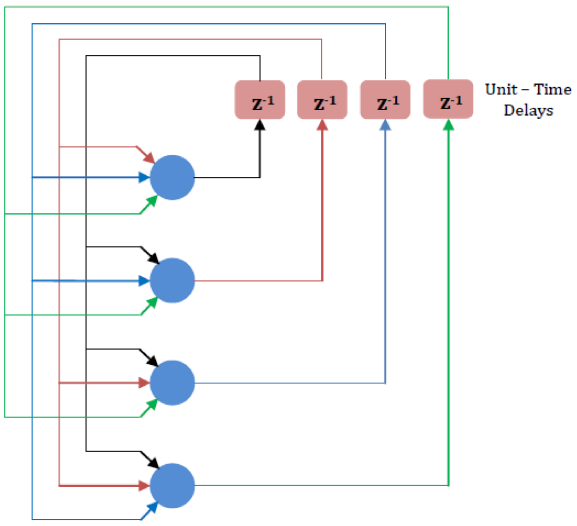

Figure 6. Simple recurrent network structure with one layer with feedback from output to the inputs and without self-feedback.

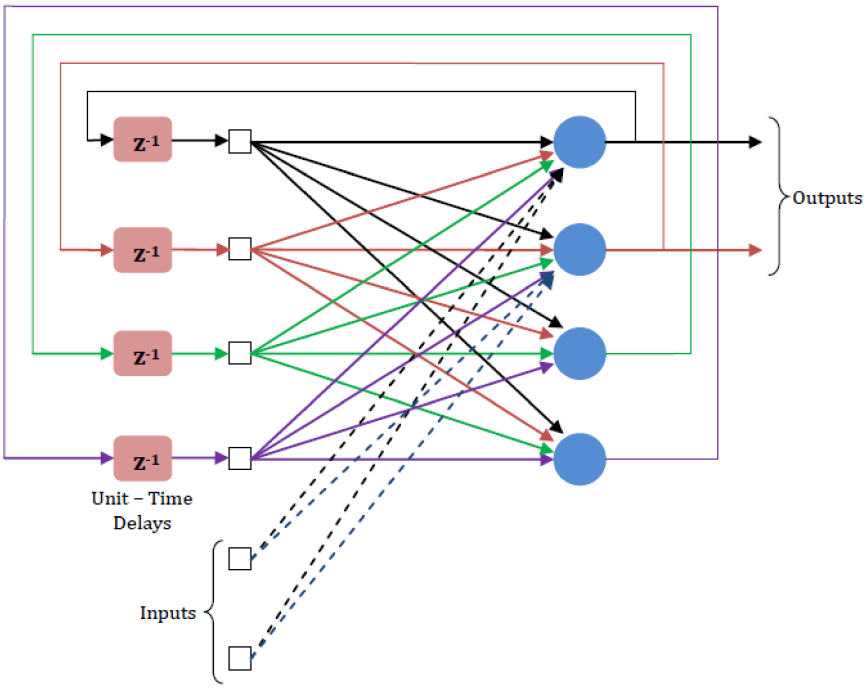

Figure 7. Recurrent network structure with hidden neurons.

Here, the case of multilayer feedforward neural networks is selected for the solution of the problem of the operating condition of power transformers through the use of key gases measured in the oil as the inputs of the neural network. The multilayer feedforward neural network will be referred to as MLNN for the rest of this paper. The structure of MLNN presented in Figure 5 makes use of Equations (1) and (2) for each one of the neurons and layer. The inputs of each neuron in each layer are multiplied by a weight factor applied on the transition from the previous layer, and there is a sum for all these multiplications inside each neuron. The result is fed in the activation function as a variable while the output feeds the inputs of the next layer and so on. The weight values of an MLNN are not random, but are computed through an important feature of the performance of MLNN procedure called training. For the description of the basic operation of the training procedure, it is of great importance to present a notation about the characterization of nodes for the input; hidden and output layers; and about the notation of the weights. For this aim, a general structure of MLNN is shown in Figure 8 with the notation of neurons and weights. The weights can also be written in matrix form, shown in Equation (3).

The training procedure is perhaps the most critical point for the development of MLNNs. The basic idea of the training procedure is feeding the network with inputs, for which the output is known, and comparison of the calculated output of the network to the real output. The percentage of correct comparisons shows the performance of the network design. In this context, the existence of training datasets of the form [inputs, outputs] is of critical importance. The computation of the performance of MLNNs comes through loss functions. Table 2 presents the most well-known loss functions. The quantifying loss function measures the cost incurred from incorrect predictions, the empirical loss function measures the total loss over the entire dataset, the binary cross entropy loss is used with models that their output is a probability between 0 and 1, while the mean squared error loss can be used with regression models whose outputs are continuous and real numbers.

$$w = \begin{bmatrix} w_{11}^{(1)} & \dots & w_{11}^{(0)} \\ \dots & \dots & \dots \\ w_{ml}^{(1)} & \dots & w_{rk}^{(0)} \end{bmatrix} \quad (3)$$

where $w_{ij}^{(h)}$ is the weight of the transition from neuron i of the h-ist layer to neuron j of the $(h+1)$-ist layer. The number of neurons can differ from layer to layer and so the matrix w is not an orthogonal matrix. However, for the simplicity of mathematics, in the points of the matrix without weights the matrix is filled with zeros.

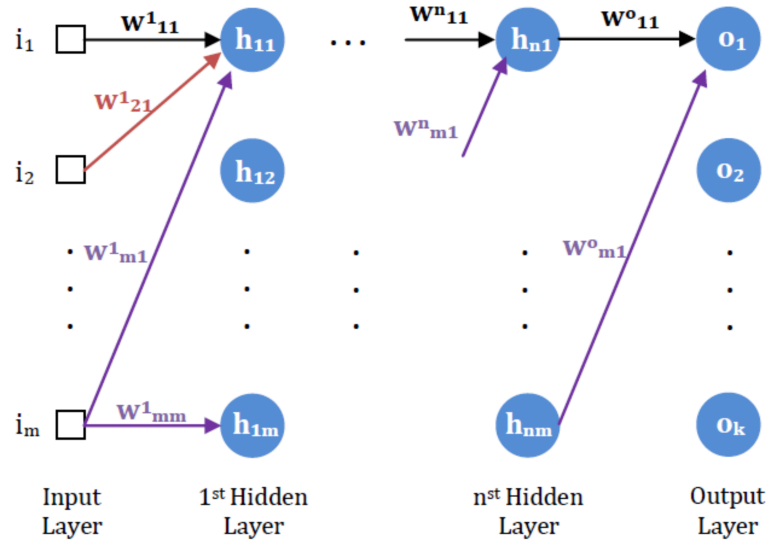

Figure 8. General structure MLNN with weights' notation.

Independently of the loss function use during the training procedure, the main aim during the design phase of NNs is the minimum loss, or in other words, loss optimization. This final aim is the result after appropriate selection of the network weights, while the minimum of the loss function can be found after the calculation of its first derivative. To achieve this optimization, the initial weight values must be known. For this reason, the training procedure starts with some random weight values for a specific MLNN structure (a specific number of layers and neurons per layer). Then, the process of training starts repeatedly until the calculated outputs approach the real outputs for all the training datasets. The algorithm generally applied consists of the following five basic steps:

- Initialize weights randomly (for example $\sim N(0, \sigma^2)$)
- Loop until convergence
- Compute the gradient $\frac{\partial J(W)}{\partial W}$
- Update the weights according to the equation:

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W} \tag{4}$$

- Return weights

The calculation for new weight values is achieved through backpropagation calculations starting from the outputs to the inputs and applying the chain rule for the calculation of specific sub gradients and internal weights. The parameter "η" in Equation (4) is well known as the "training rate" parameter and its selection is critical for correct weight calculation and total NN performance. The selection of low learning rates has the disadvantage of slow convergence and the algorithm presented above lags due to false local minima points, while many updates of the weight values are required. On the other hand, high learning rates can lead the algorithm to unstable situations and large divergencies, as Figure 9 illustrates.

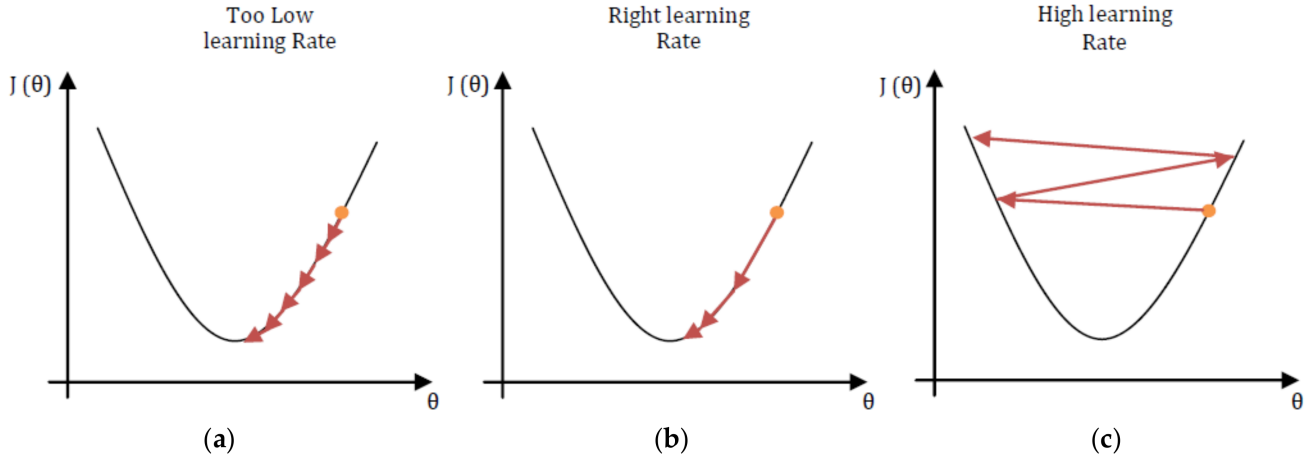

Figure 9. Effect of value for learning rate. (**a**) Low learning rate; (**b**) right learning rate; (**c**) high learning rate.

Some neural network terms that must be well understood for the correct weight selections are epochs, iterations, and batches. As already mentioned, in NN the network learns by reading the input datasets and making calculations through the entire structure. Nevertheless, this process does not take place once, but repeatedly. An epoch is the part of the learning cycle in which the trained NN uses all the training data once. This means that a full NN training process consists of a few epochs. The training data in each epoch are partitioned into packets several times. This number of separate partitions is called "batches" and each batch consists of several training samples. An increase in epochs does not guarantee better network performance.

Epoch selection is critical for the correct fitting of the input datasets. Wrong epoch number selection can lead to overfitting or underfitting problems as Figure 10 presents. By choosing a small number of epochs the NN leads to underfitting and the NN performance is extremely low. On the contrary, when the epoch number is high, it leads to overfitting, where the network is perfectly trained to recognize the training data but faces low performance with test data different from the data encountered in training. Thus, during the training process the monitoring of the number of epochs versus the loss or the accuracy is significant, as Figure 11 illustrates, and the knee of the curve is the ideal time to conclude the training process.

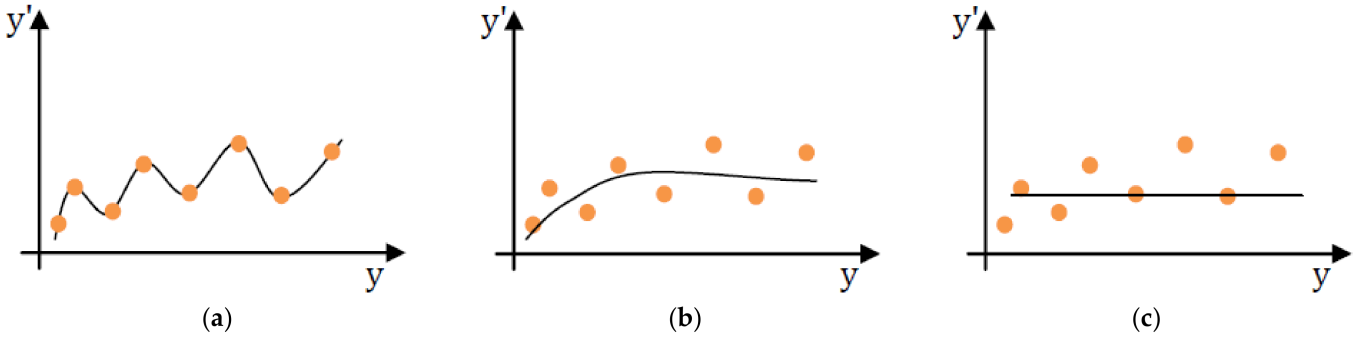

Figure 10. Fitting problems due to the selection of bias values. (**a**) Overfitting; (**b**) ideal fitting; (**c**) underfitting.

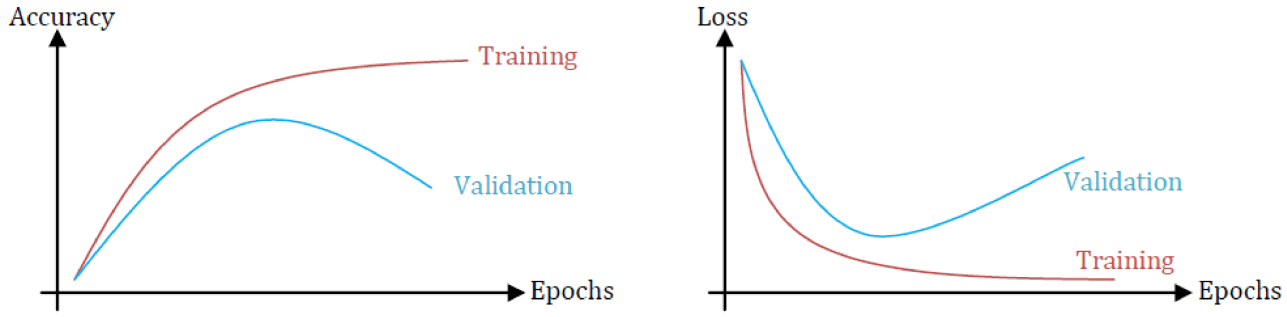

Figure 11. Accuracy and loss versus the epochs numbers.

3. Algorithm Description

This effort focuses on optimization of the neural network structure in terms of the maximum number of layers and neurons per layer, at the same time keeping high prediction performance for the problem of the operating condition of oil-immersed power transformers, using the dissolved gases as the NN inputs. To achieve this, an adaptive algorithm is proposed. Figure 12 presents the block diagram for this adaptive algorithm, which is consisted of three main parts. The philosophy of the algorithm is to select the neuron number of a specific layer and then increase the hidden layers by one and select a new neuron number for the newly inserted hidden layer. The process stops when the algorithm computes minimized error in comparison to the previously tested cases. Initially, selection for the number of hidden layers and the number of nodes per layer is conducted. The initial number of hidden layers is selected as one while the corresponding number of the neurons for this hidden layer is selected as five because the number of the inputs for the hidden layer (the gases in the oil) is also five. In the next part, for a specific number of hidden layers, the code runs the NN efficiency calculation by changing only the last layer's neurons number to achieve lower error. This is done by calculating the NN efficiency, for a specific number of nodes for the last hidden layer, many times until the best error is achieved. The philosophy behind the search of best error rate is by comparing the efficiency after each algorithm iteration with the corresponding efficiency of the previous iteration. After the best error calculation for the specific NN architecture has been calculated, the algorithm compares the efficiency with the case before the last layer's node number increase. If the error is increased, then the algorithm retains the previous number of last layer's node number. If it is achieved, then the algorithm compares the efficiency of the NN architecture with the previous case of hidden layers number. If efficiency is worst, then the algorithm keeps the previous case of NN architecture as best and terminates. An important note is that for a specific number of neurons for the last layer the code calculates NN efficiency for the entire NN structure multiple times, because MatLab which is used for the tests, initializes the weights randomly each time. Finally, the code increases the hidden layer number by one if the error is better than the previous case of the number of hidden layers and runs part two again. The code terminates when the best number of hidden layers and neurons per layer has been selected.

The proposed adaptive algorithm takes into consideration the transfer function type and the training algorithm. It uses five basic transfer functions:

- Symmetric sigmoid
- Logarithmic sigmoid
- Elliot sigmoid
- Soft max
- Linear

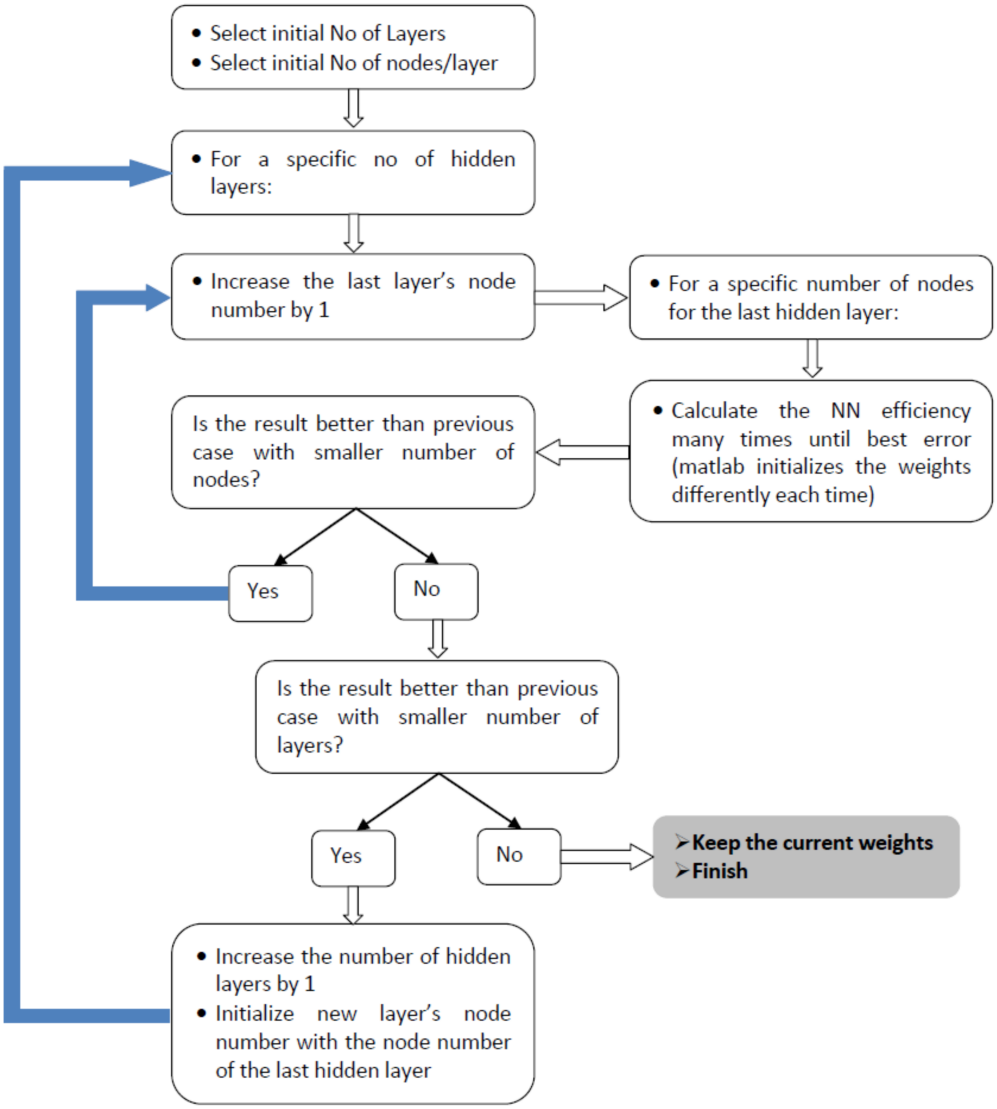

Figure 12. The adaptive algorithm block diagram.

For each case of transfer function, the proposed algorithm runs for twelve different training algorithms. These training algorithms are:

- ➢ Levenberg-Marquardt
- ➢ Bayesian Regularization
- ➢ BFGS Quasi-Newton
- ➢ Resilient Backpropagation
- ➢ Scaled Conjugate Gradient
- ➢ Conjugate Gradient with Powell/Beale Restarts
- ➢ Fletcher-Powell Conjugate Gradient
- ➢ Polak-Ribiére Conjugate Gradient
- ➢ One Step Secant
- ➢ Variable Learning Rate Gradient Descent
- ➢ Gradient Descent with Momentum
- ➢ Gradient Descent

4. Results

The proposed algorithm aims to develop an entirely new MLNN structure in an adaptive way, as described in the previous section. This MLNN will be capable of analyzing

faults in power transformers and categorizing them as mentioned in the introduction. For this reason, typical values of the key gases and the corresponding power transformer faults have been used. The database used is provided by IEEE [19]. For this reason, an algorithm described in Figure 12 is implemented in the Matlab software environment. The algorithm tries to achieve an MLNN with the minimum number of layers and the minimum number of nodes while simultaneously exhibiting high prediction efficiency. The code also examines this algorithm with five different transfer functions and for each transfer function examines twelve training algorithms. In total, 60 cases are tested, and the results for the prediction accuracy are presented in Table 3 and Figure 13. The training algorithms are presented with acronyms, and the training's algorithm meaning is presented in Table 4. Table 5 presents, for each one of the tested cases, the number of hidden layers and the number of neurons for each layer in the form "h/(n1, n2, . . .)", where h represents the number of hidden layers, n1 represents the number of neurons for the first hidden layer, n2 represents the number of neurons for the second hidden layer, and so forth.

Table 3. Prediction efficiency in percentage for each tested case. Grey shade indicates the highest prediction efficiencies.

Training Algorithms	Transfer Functions				
	Symmetric Sigmoid	Logarithmic Sigmoid	Elliot Sigmoid	Soft Max	Linear
trainbr	86.6%	58.2%	58.2%	40.3	44.3
trainlm	72.6%	78.6%	47.8%	72.6	48.3
trainbfg	34.3%	38.3%	37.3%	32.3	43.3
trainrp	34.3%	32.3%	31.8%	39.8	33.3
trainscg	42.3%	41.8%	32.3%	28.9	37.8
traincgb	37.3%	43.3%	35.8%	38.3	41.3
traincgf	35.3%	40.3%	38.3%	38.3	35.8
traincgp	34.3%	39.8%	31.8%	33.8	40.8
trainoss	39.8%	30.8%	30.3%	26.9	33.8
traingdx	33.8%	29.3%	26.9%	26.9	32.8
traingdm	28.4%	25.4%	30.3%	27.4	28.4
traingd	28.4%	29.4%	27.9%	27.9	35.8

Table 4. Explanation of training algorithm acronyms.

Training Algorithm Acronym	Training Algorithm
trainbr	Bayesian Regularization
trainlm	Levenberg-Marquardt
trainbfg	BFGS Quasi-Newton
trainrp	Resilient Backpropagation
trainscg	Scaled Conjugate Gradient
traincgb	Conjugate Gradient with Powell/Beale Restarts
traincgf	Fletcher–Powell Conjugate Gradient
traincgp	Polak–Ribiére Conjugate Gradient
trainoss	One Step Secant
traingdx	Variable Learning Rate Gradient Descent
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent

From Table 3, it is clear that the more efficient structure uses the symmetric sigmoid transfer function with Bayesian Regularization as the training algorithm with 86.6% prediction efficiency, with two hidden layers and with five neurons for each layer. The next case uses the Logarithmic Sigmoid transfer function with the Levenberg-Marquardt training algorithm. The prediction efficiency is 78.6% but uses one hidden layer with five neurons. The use of Soft Max as a transfer function in combination with the Levenberg-Marquardt training algorithm seems to have the same prediction efficiency as the combination of the Symmetric Sigmoid transfer function and Levenberg–Marquardt training algorithm, but

the case with the Soft Max transfer function uses two hidden layers with five neurons each, in contrast to the Symmetric Sigmoid where only one hidden layer with five neurons achieves the same efficiency. It is also significant to note that the case of Levenberg–Marquardt training algorithm achieves satisfactory results for the majority of transfer functions (see Table 6).

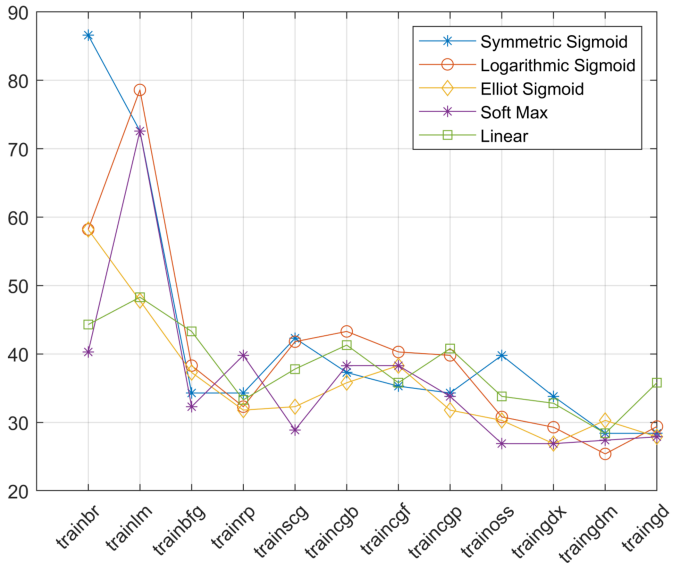

Figure 13. Prediction efficiency in percentage for each tested case.

Table 5. The number of hidden layers and the number of neurons for each hidden layer in the form "h/(n1, n2)", where h is the number of hidden layers and n1 and n2 are the number of neurons for the first and second layer. Grey shade indicates the highest prediction efficiencies.

Training Algorithms	Transfer Functions				
	Symmetric Sigmoid	Logarithmic Sigmoid	Elliot Sigmoid	Soft Max	Linear
trainbr	2/(5,5)	1/(5)	1/(5)	1/(5)	2/(5,5)
trainlm	1/(5)	1/(5)	1/(5)	2/(5,5)	1/(5)
trainbfg	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
trainrp	1/(5)	1/(5)	1/(5)	2/(5,5)	1/(5)
trainscg	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
traincgb	1/(5)	1/(5)	1/(5)	1/(5)	2/(5,5)
traincgf	2/(5,5)	1/(5)	2/(5,5)	1/(5)	1/(5)
traincgp	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
trainoss	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
traingdx	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
traingdm	1/(5)	1/(5)	1/(5)	1/(5)	1/(5)
traingd	1/(5)	1/(5)	1/(5)	1/(5)	2/(5,5)

Table 6. Comparisons between the best efficiencies over best NN architecture for the tested training algorithms and transfer functions.

Training Algorithms	Transfer Functions					
	Symmetric Sigmoid		Logarithmic Sigmoid		Soft Max	
	Efficiency	NN Architecture	Efficiency	NN Architecture	Efficiency	NN Architecture
trainbr	86.6%	2/(5,5)	58.2%	1/(5)	40.3%	1/(5)
trainlm	72.6%	1/(5)	78.6%	1/(5)	72.6%	2/(5,5)

5. Conclusions

The correct selection of the structure of MLNNs in terms of the number of hidden layers and number of neurons is a difficult procedure. This paper presented the selection of the above-mentioned parameters for the special problem of fault diagnosis in power transformers. The arising structure's efficiency keeps in touch with the literature but with the main advantage of automatically calculating the values for the number of hidden layers and neurons. The final selection of these numbers is a critical time-consuming problem to solve for the engineers before using neural networks. A first approach for an adaptive algorithm for the construction of an MLNN was presented. The algorithm aims to optimize the number of hidden layers and the number of neurons in each layer retaining the best efficiency. In the experimental part, 60 cases were tested, with 5 different transfer functions and 12 different training algorithms. From the results, it is clear that if the aim is good prediction efficiency with low requirements in total structure, then the combination of the Symmetric Sigmoid transfer function with the Levenberg–Marquardt training algorithm is ideal. The Symmetric Sigmoid transfer function with Bayesian Regularization as the training procedure, on the other hand, seems to operate well if prediction accuracy is the goal and overall MLNN structure is unimportant. The suggested adaptive approach may be utilized to generate future solutions for increasingly challenging situations and can help with the structural limitation when the NN is running on an FPGA.

Author Contributions: Conceptualization, D.A.B., C.S.P. and S.D.K.; methodology, D.A.B.; software, D.A.B.; validation, K.K.K., C.S.P. and G.C.I.; formal analysis, D.A.B. and C.S.P.; investigation, D.A.B., S.D.K. and C.S.P.; resources, D.A.B. and S.D.K.; data curation, D.A.B. and C.S.P.; writing—original draft preparation, D.A.B., K.K.K. and G.C.I.; writing—review and editing, K.K.K. and G.C.I.; visualization, D.A.B.; supervision, C.S.P.; project administration, C.S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs. Publicly available datasets were analyzed in this study. This dataset can be found here: https://ieee-dataport.org/open-access/botnet-dga-dataset.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aziz, S.; Peng, J.; Wang, H.; Jiang, H. Admm-based distributed optimization of hybrid mtdc-ac grid for determining smooth operation point. *IEEE Access* **2019**, *7*, 74238–74247. [CrossRef]
2. US. Department of the Interior Bureau of Reclamation. *Transformers: Basics, Maintenance, and Diagnostics*; U.S. Department of the Interior Bureau of Reclamation: Denver, CO, USA, 2005.
3. Cigre Working Group. *Guide for Transformer Maintenance—Cigre Working Group A2.34*; Cigre: Paris, France, 2001; ISBN 978-2-85873-134-3.
4. Kanno, M.; Oota, N.; Suzuki, T.; Ishii, T. Changes in ECT and dielectric dissipation factor of insulating oils due to aging in oxygen. *IEEE Trans. Dielectr. Electr. Insul.* **2001**, *8*, 1048–1053. [CrossRef]
5. Aslam, M.; Arbab, M.N.; Basit, A.; Ahmad, T.; Aamir, M. A review on fault detection and condition monitoring of power transformer. *Int. J. Adv. Appl. Sci.* **2019**, *6*, 100–110.
6. Patel, D.M.K.; Patel, D.A.M. Simulation and analysis of dga analysis for power transformer using advanced control methods. *Asian J. Converg. Technol.* **2021**, *7*, 102–109. [CrossRef]
7. Siva Sarma, D.V.S.S.; Kalyani, G.N.S. ANN approach for condition monitoring of power transformers using DGA. In Proceedings of the IEEE Region 10 Conference TENCON, Chiang Mai, Thailand, 24 November 2004. [CrossRef]
8. Hashemnia, N.; Islam, S. Condition assessment of power transformer bushing using SFRA and DGA as auxiliary tools. In Proceedings of the IEEE International Conference on Power System Technology (POWERCON), Wollongong, NSW, Australia, 28 September–1 October 2016. [CrossRef]
9. Nemeth, B.; Laboncz, S.; Kiss, I. Condition monitoring of power transformers using DGA and Fuzzy logic. In Proceedings of the IEEE Electrical Insulation Conference, Montreal, QC, Canada, 31 May–3 June 2009. [CrossRef]
10. Aciu, A.-M.; Nicola, C.-I.; Nicola, M.; Nițu, M.-C. Complementary Analysis for DGA Based on Duval Methods and Furan Compounds Using Artificial Neural Networks. *Energies* **2021**, *14*, 588. [CrossRef]

11. Chatterjee, A.; Roy, N. Health Monitoring of Power Transformers by Dissolved Gas Analysis using Regression Method and Study the Effect of Filtration on Oil. *Int. J. Electr. Comput. Eng.* **2009**, *3*, 1903–1908.
12. Dhini, A.; Faqih, A.; Kusumoputro, B.; Surjandari, I.; Kusiak, A. Data-driven Fault Diagnosis of Power Transformers using Dissolved Gas Analysis (DGA). *Int. J. Technol.* **2020**, *11*, 388–399. [CrossRef]
13. Papadopoulos, A.E.; Psomopoulos, C.S. The contribution of dissolved gas analysis as a diagnostic tool for the evaluation of the corrosive sulphur activity in oil insulated traction transformers. In Proceedings of the 6TH IET Conference on Railway Condition Monitoring (RCM), University of Birmingham, Birmingham, UK, 17–18 September 2014.
14. Papadopoulos, A.E.; Psomopoulos, C.S.; Kaminaris, S.D. Evaluation of the insulation condition of the two ONAN transformers of PUAS. In Proceedings of the International Scientific Conference eRA-10, Piraeus, Greece, 23–25 September 2015; pp. 1–7.
15. Papadopoulos, A.; Psomopoulos, C.S. Dissolved Gas Analysis for the Evaluation of the Corrosive Sulphur activity in Oil Insulated Power Transformers. In Proceedings of the 9th Mediterranean Conference and Exhibition on Power Generation, Transmission and Distribution, IET (MedPower 2014), Athens, Greece, 2–5 November 2014.
16. Huang, Y.-C.; Yang, H.-T.; Huang, C.-L. Developing a new transformer fault diagnosis system through evolutionary fuzzy logic. *IEEE Trans. Power Deliv.* **1997**, *12*, 761–767. [CrossRef]
17. Aziz, S.; Wang, H.; Liu, Y.; Peng, J.; Jiang, H. Variable Universe Fuzzy Logic-Based Hybrid LFC Control With Real-Time Implementation. *IEEE Access* **2019**, *7*, 25535–25546. [CrossRef]
18. Zhang, R.; Li, G.; Bu, S.; Aziz, S.; Qureshi, R. Data-driven cooperative trading framework for a risk-constrained wind integrated power system considering market uncertainties. *Int. J. Electr. Power Energy Syst.* **2023**, *144*, 108566. [CrossRef]
19. IEEE DataPort. Available online: https://ieee-dataport.org/documents/dissolved-gas-data-transformer-oil-fault-diagnosis-power-transformers-membership-degree#files (accessed on 6 June 2022).
20. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Prentice Hall: Prentice, NJ, USA, 1995.
21. Paluszek, M.; Thomas, S. *MATLAB Machine Learning*; Apress: Newark, NJ, USA, 2017.
22. Mathworks. Available online: https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/machine-learning-resource/machine-learning-with-matlab.pdf (accessed on 10 July 2022).
23. Sarajcev, P.; Kunac, A.; Petrovic, G.; Despalatovic, M. Artificial Intelligence Techniques for Power System Transient Stability Assessment. *Energies* **2022**, *15*, 507. [CrossRef]
24. Ambare, V.; Uphad, Y.; Tikar, S.; Rabade, N.; Ingle, N. Literature Review on Artificial Intelligence. *Int. J. Future Revolut. Comput. Sci. Commun. Eng.* **2019**, *5*, 1–4.
25. Haykin, S. *Neural Networks and Learning Machines*, 3rd ed.; Pearson Prentice Hall: Prentice, NJ, USA, 2008.
26. Nanni, L.; Brahnam, S.; Paci, M.; Ghidoni, S. Comparison of Different Convolutional Neural Network Activation Functions and Methods for Building Ensembles for Small to Midsize Medical Data Sets. *Sensors* **2022**, *22*, 6129. [CrossRef] [PubMed]
27. Ding, B.; Qian, H.; Zhou, J. Activation functions and their characteristics in deep neural networks. In Proceedings of the Chinese Control And Decision Conference (CCDC), Shenyang, China, 9–11 June 2018. [CrossRef]