Article

Optimization of the ANNs Predictive Capability Using the Taguchi Approach: A Case Study

Andrea Manni [1,2], Giovanna Saviano [2] and Maria Grazia Bonelli [3,4,*]

1 Chemical Research 2000 S.r.l., Via S. Margherita di Belice 16, 00133 Rome, Italy; info@cr2000.it or andrea.manni@uniroma1.it
2 Department of Chemical, Materials and Environmental Engineering (DICMA), "La Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy; giovanna.saviano@uniroma1.it
3 Programming and Grant Office Unit (UPGO), Italian National Research Council (CNR), Piazzale Aldo Moro 7, 00185 Rome, Italy
4 InterUniversity Consortium Georesources Engineering (CINIGeo), Corso Vittorio Emanuele II 244, 00186 Rome, Italy
* Correspondence: mariagrazia.bonelli@cnr.it

Abstract: Artificial neural networks (ANNs) are a valid alternative predictive method to the traditional statistical techniques currently used in many research fields where a massive amount of data is challenging to manage. In environmental analysis, ANNs can analyze pollution sources in large areas, estimating difficult and expensive to detect contaminants from other easily measurable pollutants, especially for screening procedures. In this study, organic micropollutants have been predicted from heavy metals concentration using ANNs. Sampling was performed in an agricultural field where organic and inorganic contaminants concentrations are beyond the legal limits. A critical problem of a neural network design is to select its parametric topology, which can prejudice the reliability of the model. Therefore, it is very important to assess the performance of ANNs when applying different types of parameters of the net. In this work, based on Taguchi L_{12} orthogonal array, turning experiments were conducted to identify the best parametric set of an ANNs design, considering different combinations of sample number, scaling, training rate, activation functions, number of hidden layers, and epochs. The composite desirability value for the multi-response variables has been obtained through the desirability function analysis (DFA). The parameters' optimum levels have been identified using this methodology.

Keywords: artificial neural network; Design of Experiment (DoE); parametric design; forecasting; environmental pollution

check for updates

Citation: Manni, A.; Saviano, G.; Bonelli, M.G. Optimization of the ANNs Predictive Capability Using the Taguchi Approach: A Case Study. *Mathematics* **2021**, *9*, 766. https://doi.org/10.3390/math9070766

Academic Editor: Daniel-Ioan Curiac

Received: 23 February 2021
Accepted: 29 March 2021
Published: 1 April 2021

1. Introduction

Artificial neural networks (ANNs) are a machine learning technique that is widely used as an alternative forecasting method to traditional statistical approaches in many scientific disciplines [1], such as marketing, meteorology, finance, or environmental research, when a massive amount of data is challenging to manage.

Unlike conventional methodologies, ANNs are data-driven self-adaptive, and non-linear methods that do not require specific assumptions about the underlying model [2]. ANNs are data-driven because they can process the available data (input) and produce a target variables vector (output) through a feed-forward algorithm. Moreover, the method is self-adaptive. In fact, a neural network can approximate a wide range of statistical models without hypothesizing in advance on any relationships between the dependent and independent variables. Instead, the form of the relationship is determined during the network learning process. If a linear relationship between the dependent and independent variables is appropriate, the neural network results should closely approximate those of the linear regression model. If a non-linear relationship is more appropriate, the neural network will automatically match the "correct" model structure. Therefore, this algorithm is suitable for approximating complex relationships between the input and output variables with a non-linear optimization [3].

ANNs algorithms simulate how the human brain processes information through the nerve cells, or neurons, connected to each other in a complex network, within a computational model [4]. The similarity between an artificial neural network and a biological neural network relies on the network's acquirement of knowledge through a "learning process" [5]. From the initial data (input), it is possible to determine the target variable (output) through the complex system of cause–effect relationships that the model discovers.

The neuron (node) is the primary processing unit in neural networks. Each artificial neuron has weighted inputs, transfer functions, and one output. An example of neural propagation is the McCulloch–Pitts model (Figure 1), which combines its inputs (typically as a weighted sum) to generate a single output.

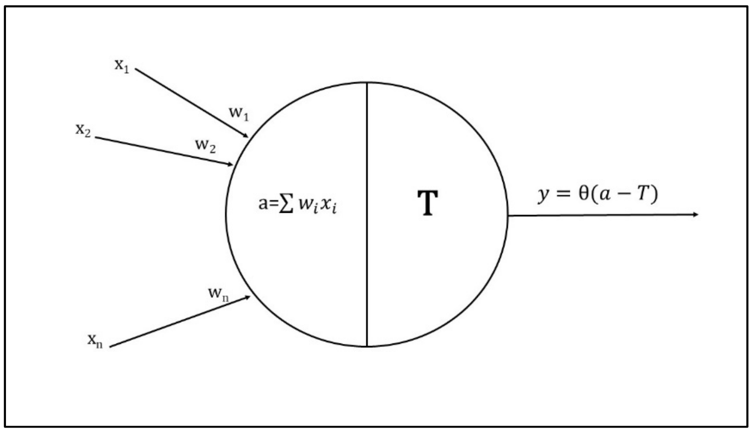

Figure 1. Example of the model of neural propagation.

In this model, the inputs x_i are multiplied by the weights w_i, with w_0 as bias. Bias shifts the activation function by adding a constant to the input to better fit the data prediction. Bias in neural networks can be thought of as analogous to the role of a constant in a linear function, whereby the constant value transposes the line: without w_0, the line would go through the origin (0, 0), and the fit could be poor. If the weighted sum of the inputs "a" overcomes a threshold value T, neurons release its output y, which is a function of (a-T). In other words, the arriving signals (inputs), multiplied by the connection weights, are first summed and then passed through an activation function (θ) to produce the output [6]. A unit feeds its output to all the nodes on the next layer, but there is no feedback to the previous layer (feed-forward network). Weights represent the system memory and indicate the importance of each input neuron in the output generation processing. The activation function is used to introduce non-linearity in the network's modeling capabilities [7]. Activations are typically a number within the range of 0 to 1, and the weight is a double, e.g., 2.2, −1.2, 0.4.

In a neural network, there are many parameters for building a model. It is possible to change the number of layers, the number of neurons for each layer, the type activation function to use in each layer, the training duration, and the learning rule.

The optimization of neural network architecture is a reccurring topic of several studies. Generally, the different approaches regard empirical or statistical methods to analyze the effect of an ANN's internal parameters, choosing the best values for them based on the model's performance [8,9], e.g., through the notions from Taguchi's design of experiments [10], or hybrid methods composed of a feed-forward neural network and an optimization algorithm to maximize weights and biases of the network [11,12]. Other authors have compared the performance of different feed-forward and recurrent ANNs architectures, using the trial-and-error method to determine these parameters' optimal choice [13,14]. For each of the architectures, the activation functions, the number of layers, and the number of neurons are found employing a trial-and-error procedure, as the best compromise between estimation accuracy, time of training, and the need to avoid training overfitting. A different optimization approach adds and/or removes neurons from an initial architecture using a previously specified criteria to indicate how the ANN performance is affected by the changes [15]. The neurons are added when training is slow or when the mean squared error is larger than a specified value. Simultaneously, the neurons are removed when a change in a neuron's value does not correspond to a change in the network's response or when the weight values associated with this neuron remain constant for a large number of training epochs. Several studies have proposed optimization strategies by varying the number of hidden layers and hidden neurons through the application of genetic operators (GA) and evaluation of the different architectures according to an objective function [16,17]. This approach considers the problem as a multi-objective optimization, and the solution space is the collection of all the different combinations of hidden layers and hidden neurons. Given a complex problem, a genetic algorithm (GA) is developed to search the solution space for the "best" architectures, according to a set of predefined criteria. A GA is an artificial intelligence search metaheuristic derived from the process of biological organism evolution [18]. The GA is often preferred to conventional optimization algorithms due to its simplicity and high performance in finding the solutions for complex high-dimensional problems. The ability to handle arbitrary objectives and constraints is one of the advantages of the genetic algorithm approach [19]. Unlike the traditional optimization method, GA uses probabilistic transition rules instead of using deterministic rules. It works by coding the solution instead of using the solution itself. Moreover, it works with a population of solutions instead of a single solution. However, even if GA requires less information about the problem, designing an objective function and getting the proper representation can be difficult. In addition, GA does not constantly assess a global optimum, and it can produce a quick response time only in the case of a real-time application [20].

In this study, because of the large number of variables and factors to optimize, despite the advantages of GA as an optimization method, it has been preferred to take a strategy of reduction of the number of experiments to conduct, using Taguchi's factorial Design of Experiment (DoE) to identify the best architecture (number of hidden layers, number of hidden neurons, choice of input factors, training algorithm parameters, etc.) of a Multi-Layer Perceptron model, considering as a case of study an environmental problem of pollution characterization. Taguchi's method is an important tool for robust DoE. It represents a simple and systematic approach to optimize the design, maximizing performance and quality, and minimizing the number of experiments, reducing the experimental design cost [21]. The main reason to choose the Taguchi method instead of other optimization algorithms (such as GA) is its capability to reduce the time required for experimental investigation and to study the influence of individual factors to determine which factor has more influence and which has less. Orthogonal arrays, the principal tool on which the method is based, accommodate many design factors simultaneously, obtaining different information for each test, even when applying the most straightforward deterministic statistical techniques. In this paper, the optimization of a shallow network (§ 2.1.1) has been considered. Many authors have implemented the Taguchi method also for optimization of a deep neural network (DNN) [22,23]. A DNN is a net with multiple layers between the input and output layers. Deep learning has demonstrated an excellent performance to solve pattern recognition problems, such as computer vision, speech recognition, natural language processing, and brain–computer interface [24]. At the same time, it is less often used as a forecasting method.

2. Materials and Methods

2.1. The Multi-Layer Perceptron (MLP) Model

Among the various types of ANNs, this work focuses on the Multi-Layer Perceptron (MLP) model with a feed-forward back-propagation learning rule based on a supervised procedure. The information moves only in one direction—forward—from the input nodes, through the hidden nodes, and to the output nodes. There are no cycles or loops in the network. It is a particular type of fully connected network with three or more layers (an input and an output layer with one or several hidden layers) of non-linearly activating nodes. The following figure illustrates the concept of a three-layer MLP with m input nodes, h hidden nodes, and t output nodes (Figure 2) [25].

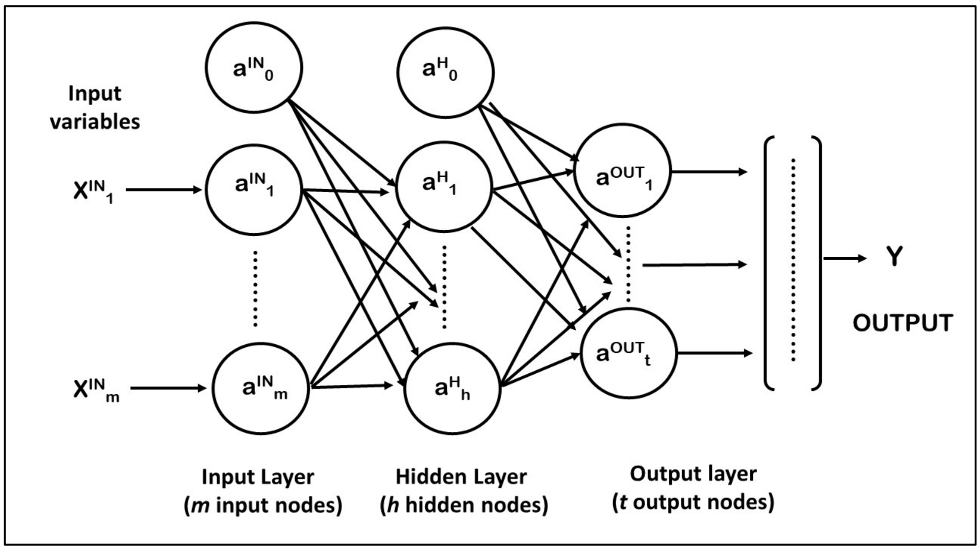

Figure 2. A Three-layer Multi-Layer Perceptron (MLP), with m input nodes (IN), h hidden nodes (H), and t output nodes (OUT). The term $a_0^{(J)}$ is the bias, for J = IN, H

2.1.1. Neurons and Layers

Therefore, the general MLP architecture consists of multiple layers of nodes (neurons) in a directed graph, which are fully connected to each other. The neurons calculate the weighted sum of inputs and weights, add the bias, and execute an activation function.

A neural network will always have an input and output layer and zero or more hidden layers. In a simple network, the input layer is responsible for receiving the inputs, performing the calculations through its neurons, and transmitting the information to the output layer, which produces the final results through its neurons.

In a complex neural network with one or multiple hidden layers, the output layer receives inputs from the previous hidden layers. Each hidden layer contains the same number of neurons and is not visible to the external systems but "private" to the neural network [26].

The number of hidden layers is challenging to decide. A network with only one hidden layer is defined as a shallow network. It is possible to add an arbitrary number of hidden layers to the MLP to create a deep network. It is possible to add a random number of hidden layers to the MLP to create a deep network (i.e., convolutional neural networks (CNNs)), which have several hidden layers, often of various types [27]. A deeper architecture can increase the precision with which a function can be approximated on a fixed set of parameters and improve the generalization after the learning process. The main explanation is that a deep model can extract the input and output variables' features better than shallow models. However, one hidden layer often shows sufficient predictive capability for many different chemometrics phenomena [28].

The number of neurons in the individual layer and the number of hidden layers can influence the prediction abilities of the MLP. Training a neural network on a large dataset takes a lot of time. It can depress the generalizing capabilities of the MLP through overfitting or memorization [29], but too little data could hinder the learning process, as not enough information is available.

2.1.2. Activation Function

How the network connects the inputs to the target variables through the hidden layers depends on the activation function choice. The activation functions are chosen based on the kind of available data (binary, bipolar, decimal, etc.) and the type of layer., while non-linear sigmoid functions are used in the hidden layers (usually the hyperbolic tangent function). There is no universally valid rule to define the various layers activation function, and many different options are available [30]. The identity function is almost always used in the input layer. A linear activation function does not help with the complexity or various parameters of usual data that is fed to the neural networks.A continuous non-linear function in hidden layers is generally preferred because the relationships between hidden nodes and output are non-linear [31]. No matter how many layers are in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function), and all layers of the neural network collapse into one [32].

The most used non-linear activation functions of the hidden nodes are the logistic, the sigmoid, and the hyperbolic tangent. This is not an exhaustive list of activation functions used for hidden layers, but they are the most commonly used. The activation function of the output units is the identity function, the logistic, and the sigmoid. Therefore, it is possible to choose a linear function (identity function) in the last layer unless the network has no hidden layer and a single output node [33].

2.1.3. The Training Process and Pre-Treatment of Data

As mentioned above, MLP has a back-propagation learning rule: the network learns by calculating the errors of the output layer to find the errors in the hidden layers [34]. Learning (training) a network is the process of adapting or modifying the connection weights between neurons so that the network can fulfill a specific task. Back-propagation is the most common learning algorithm, which is an iterative gradient descent algorithm. Back-propagation algorithms are known for their ability to learn, and they are highly suitable for problems in which no relationship is found between input and output [35].

The MLP learning process occurs in the following consecutive phases:

1. Training phase, to find the weights that represent the relationship between ANN inputs and outputs.
2. Testing phase, to optimize the weights and to estimate model accuracy by error indicators.

Before the training, the original dataset is usually split into an independent training set and test set. The first subset contains a known output, and the model learns from this data to be generalized to other data later. The second subset is used to test the model predictions on it.

The criteria defining the size of the training and test set are different. The subsection of data can be random, defining a priori the percentage of units to insert in each group [36]. The most used solution is 70% for training and 30% for testing, especially for large datasets [37].

Very often, the original dataset contains features highly various in measurement unit or range. It is necessary to bring all features to the same level of measurement by a scaling rule.

There are two different ways to scale a dataset. The first option is to resize the input (independent) and output (dependent) variables to [0,1] by normalization. In this case, if the sigmoid activation function at the output is used, the extreme data (outliers) could not be seen in the training set [38]. Another option is to use a standardization (z-score) with a mean of 0 and a standard deviation of 1. In that case, there are no outlier problems, and the trained network produces, in general, better results, but the advantage reduces as the network and sample size grow [39].

2.1.4. The Training Cycles and Network Performance

The MLP reads the input and output variables during a training set and optimizes the weight values through backward propagation to reduce the difference between the predicted and observed variables. The prediction error is minimized across many training cycles (epochs) until the network hits a specified accuracy level [40]. Therefore, an epoch is an entire training cycle of the original dataset. It is different from iteration; that is, the number of steps needed to complete one epoch.

Training for too many iterations could lead to overfitting, and the error on the test set starts to climb. In this case, the network could lose its ability to generalize the results [32]. Given the dataset complexity and variability, it could take from 10,000 to 60,000 epochs to get some reasonable accuracy on the test set [41].

Once the training is completed, the predictive capability must be verified because the network cannot be suitable to generalize the results unless the model had an excellent performance in training and test set.

Generalization is a critical aspect of any model construction. It is based upon some specified neural network performance measures. Since MLP is a function of input nodes that minimize the prediction error, to assess the network performance, the three most frequently performance measures utilized are as follows [42]:

(a) Coefficient of Determination (R^2)

$$R^2 = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_i - \overline{Y}_i\right)^2 - \sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}\left(Y_i - \overline{Y}_i\right)^2}}$$

(b) Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\,\hat{Y}_i - Y_i\right|$$

(c) Root Mean Squared Error ($RMSE$)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}$$

R^2 provides the variability measure of the data reproduced in the model. As this test does not give the model's accuracy, other statistical parameters have to be reported. *MAE* and *RMSE* measure residual errors, which provide a global idea of the difference between the observed and modeled values.

The minimum *RMSE* or *MAE* and the maximum R^2 are often used to select the "better" neural network [43].

RMSE is calculated in the training and test set. Comparing both values, if they are of the same order of magnitude, the neural network provides reliable predictions [44].

As already mentioned, the predictive performance of a neural network depends on several parameters. The Taguchi approach, an efficient method of Design of Experiments (DoE), will seek to find the optimal parametric set of an MLP model. Data already used in previous research about artificial neural networks in evaluating the contamination in agricultural soils have been considered as a case study.

2.1.5. The Taguchi Design of Experiments Method

Taguchi developed a methodology of experimental design to identify the most critical design variables to minimize uncontrollable factors' effects on product variations [45]. Taguchi design is used in many industrial experiments, especially in the field of quality control. This method is applied to determine an analytical experiment outcome consisting of different factors, with two or more levels, and it belongs to the factorial design class.

A full factorial design identifies all possible combinations for a given set of factors that influences a response variable. Therefore, if an experiment involves a significant number of factors, a full factorial design carries out a large number of experiments, with prohibitive time and costs (§ 3.4). A subset from all the possibilities is selected by factorial design methods to reduce the number of experiments to a workable level [46].

Taguchi constructed a particular set of general designs for factorial experiments to use in a lot of applications. This approach uses a set of arrays called orthogonal arrays that defines the minimal number of experiments that could give the complete information of all the factors that affect the outcome. Each experiment's results are converted to a signal-to-noise (*S/N*) ratio to determine the combination of control factor levels to improve the system's stability and reduce quality loss. The crucial point of this method is to choose the level combinations of the factors for each experiment.

2.2. A Case Study: Use of Artificial Neural Networks to Evaluate Organic and Inorganic Contamination in Agricultural Soils

The agricultural soils around an industrialized site are often exposed to environmental micropollutants due to the different emission sources and anthropogenic activities. In these extensive lands, the characterization of pollution is challenging to perform due to the vast size of the sample area, with a wasteful expenditure of time and money. Instead, it could be essential to provide a preliminary analysis of the site establishing the presence or absence of pollution by the combined use of the screening tools (such as a portable metal analyzer) and statistical techniques. In the following case of study, artificial neural networks have been used to estimate the concentrations of organic compounds (dioxins, furans, and PCBs) found in contaminated soil through their relationships with inorganic micropollutants present in the same area [47].

A total of 75 soil samples (64 surface soils and nine deep soils) have been analyzed for some heavy metals (Cu, As, Zn, Pb, Mn, Hg, Ni, Fe, Mg, and Cr) by Field Portable X-Ray Fluorescence (FP-XRF), dioxins and furans (PCDD/Fs), and dioxin-like PCBs, performed by High-Resolution Gas Chromatography/High-Resolution Mass Spectrometry (HRGC/HRMS).

For this application, only data from the 64-surface soil samples, which have homogeneous chemical and physical characteristics, have been used to predict dioxins and furans (PCDD/Fs) concentrations through the values of the ten heavy metals above mentioned.

All the MLP models will be performed by the package IBM-SPSS v. 25, while the statistical software JMP v. PRO 14.0.0 will be applied for Taguchi design.

The Neural Network function in IBM-SPSS can compute a set of non-linear data modeling tools consisting of input and output layers plus one or two hidden layers. The connections between neurons in each layer have associated weights, which are iteratively adjusted by the training algorithm to minimize error and provide accurate predictions [48]. It is possible to set the conditions under which the network "learns" and to control the training stopping rules and network architecture or to let the procedure automatically choose the architecture. In the latter case, SPSS divides the data into training and testing sets on a fixed percentage, generally 70% and 30%, respectively. The independent variables (input nodes) are automatically standardized. Simultaneously, the architecture comprises only one hidden layer, the hyperbolic tangent, as the activation function for hidden nodes and identity function for output nodes. The automatic duration of the training is calculated in 10 epochs.

Thus, the available data used to build a neural network by the automatic procedure have been created. The result has been an MLP feed-forward network based on 10 inputs, three hidden nodes in one layer, and one output (Figure 3).

The SPSS function has randomly divided the 64 samples into a training set of 45 units (70.3%) and a test set of 19 units (29.7%).

The training set was used to train the network and the test set was used to evaluate the prediction performance: R^2 is 0.87, and the *RMSE* and *MAE* values in the training and test set are of the same order. Therefore, the MLP model provides reliable predictions.

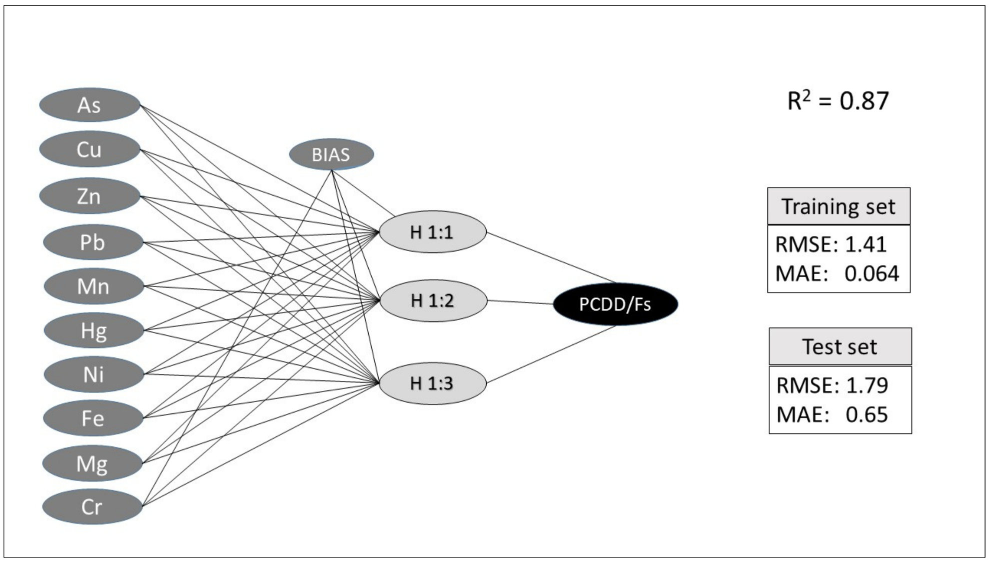

Figure 3. MLP feed-forward network 10-3-1. Input variables have been standardized. The training rate set was 70.3%. Activation functions were hyperbolic tangent for hidden nodes and identity function for output nodes. The prediction error has been minimized across ten epochs. The RMSE and MAE values in the training and test set are of the same order, even if not remarkably low, and R^2 is 0.87. The network provides reliable predictions.

The Taguchi Design Steps to Optimize the Predictive Xapability of ANNs

The MLP model mentioned in Section 3.1 was built, allowing the automatic selection of the network parameters. However, the choice of the architectural features and the stopping rules influence the training results and the model's predictive capability.

The analysis used in this study aimed to find the parameter design that determined the best performance of the MLP model through the Taguchi approach.

Generally, in industrial quality control, the Taguchi methodology includes several steps of planning, conducting, and evaluating results of a matrix of experiments to determine the best levels of control factors [49]. A flowchart of the various steps of the Taguchi method is depicted in Figure 4.

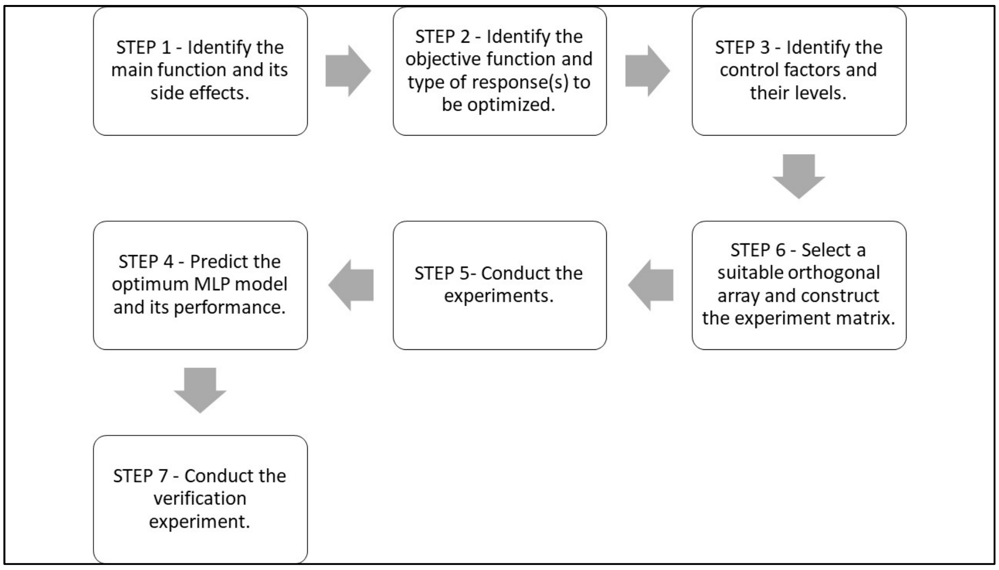
STEP 1 - Identify the main function and its side effects.
STEP 2 - Identify the objective function and type of response(s) to be optimized.
STEP 3 - Identify the control factors and their levels.
STEP 6 - Select a suitable orthogonal array and construct the experiment matrix.
STEP 5- Conduct the experiments.
STEP 4 - Predict the optimum MLP model and its performance.
STEP 7 - Conduct the verification experiment.

Figure 4. Flow chart of the Taguchi method.

3. Results

According to these steps, a series of experiments has been carried out. The procedure has been illustrated in the following subsections.

3.1. Step 1. Identify the Main Function and Its Side Effects

The optimization of ANNs' predictive capability has been considered as the main function. No side effects have been recorded. Before proceeding to the following steps, it is necessary to detect all the factors influencing the network performance, identifying them as signal (or control) and/or noise factors [50]. Signal factors are the system control inputs, while noise factors are typically difficult or expensive to control. In this analysis, all the considered factors are controllable in the model; they have been classified as "signal factors" and none were classified as "noise factors".

3.2. Step 2. Identify the Objective Function and Type of Response(s) to Be Optimized

The index R^2, one of the three performance measures (§ 2.1.4), was chosen as the objective function or response variable to be maximized. The target value of R^2 has been set to 0.70 or more. Taguchi recommends using a loss function to quantify a design's quality, defining the loss of quality as a cost that increases quadratically with the deviation from the target value. Usually, the quality loss function considers three cases: nominal-the-best, smaller-the-better, and larger-the-better [51], corresponding to three types of continuous measurable responses to be optimized [52]:

Target-is-the-best (TTB) response, when the experiment's objective is to achieve a desired target performance for the response variable.

Smaller-the-better (STB) response, in which the desired target is to minimize the response variable.

Larger-the-better (LTB) response, when the experiment's objective is to maximize the response within the acceptable design limits.

According to this scheme, in this analysis, R^2 qualifies as an LTB response.

To quantify the output quality, Taguchi suggests that the objective function to be optimized is the signal/noise ratio (*S/N* Ratio) [53], which is a logarithmic function of the response variable with a different trend depending on the type of response. An objective function of an LTB response is calculated assuming the following formula:

$$\frac{S}{N}Ratio = -10log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{y_i^2}\right)$$

where n is the number of the experiments or runs in a Taguchi design, and y is the response value in the run i. The level indicated the best experimental results obtained by the calculation of average *S/N* ratio for each factor and the optimal level of the process parameters with the largest *S/N* ratio.

3.3. Step 3. Identify the Control Factors and Their Levels

The signal (or control) factors and their levels (§ 2.1) are shown in Table 1.

Table 1. Signal (or control) factors and their levels.

Factor	Levels		
Number of samples	10	30	50
Input scaling		normalized; standardized	
Training rate (%)	60	70	80
Act. Function H		Sigmoid; Hyperbolic; tangent	
Act. Function O	identity	sigmoid	hyp.tangent
n. Hidden Layers	1	-	2
Epochs	10	10,000	60,000

A parametric set of seven variables of an MLP model has been considered and in detail listed below:

- Number of samples. The experiment can detect the minimum number of sample units sufficient for the network to learn. In a 64-sample database, three levels of this factor have been fixed: 10, 30, and 50 units.
- Input scaling rule. Two levels are corresponding to the different criteria to scale input data (§ 2.3):

(a) normalization, using the formula

$$p_i = \frac{x_i \ - \min(x)}{\max\,(x) - \min(x)}$$

(b) standardization, using the formula

$$z_i = \frac{x_i \ - \mu}{\sigma}$$

- Training rate (%): three levels of percentage have been considered for computing the size of the training set (test set): 60% (40%), 70% (30%), and 80% (20%).
- Activation function of hidden and output nodes: as mentioned above (§ 2.2), two levels have been chosen for the activation function of the hidden nodes (sigmoid and hyperbolic tangent), and three levels for the activation function of output nodes (identity function, sigmoid, and hyperbolic tangent).
- Number of hidden layers: to determine if a deep network has better predictive performance, two levels of this factor have been considered: one or two hidden layers, as allowed by Neural Network function in IBM-SPSS.
- Epochs. The training process duration has been set to three levels: 10, 10,000, and 60,000 epochs.

3.4. Step 4. Select a Suitable Orthogonal Array and Construct the Experiment Matrix

A full factorial design considers all input factors at two or more levels each, whose experimental units take on all possible combinations of these levels across all such factors. This experiment allows studying each factor effect on the response variable: if there are k factors each at two levels, a full factorial design has 2^k runs. Thus, for seven factors at two or three levels, it would require many experiments (2^7 or 3^7) to be carried out, and too many observations to be economically viable, as stated above (§ 3).

Taguchi suggested a particular method using the orthogonal array with fewer experiments to be conducted to resolve this problem. The degrees of freedom (DoF) have to be calculated, considering one DoF for the mean value to select an adequate orthogonal array, and for each factor, the number of levels less 1. Thus, the degrees of freedom of this design are 12 (Table 2), and the most suitable orthogonal array (OA) is L_{12} (Table 3):

Table 2. Degrees of freedom of Taguchi design.

Factor	**Number of Level (n_i)**	**Degrees of Freedom (n_i) −1**
Mean value	-	1
Number of samples	3	2
Input scaling	2	1
Training rate (%)	3	2
Act. Function H	2	1
Act. Function O	3	2
n. Hidden Layers	2	1
Epochs	3	2

Table 3. Orthogonal array (OA) of Taguchi design L12.

OA	Number of Samples	Input Scaling	Training Rate (%)	Act. Function H	Act. Function O	n. Hidden Layers	Epochs
L1	10	norm	60	sigm	lin	1	10
L2	30	norm	70	sigm	hp tg	1	10,000
L3	50	norm	80	sigm	sigm	1	60,000
L4	10	norm	80	hp tg	hp tg	2	10,000
L5	30	norm	60	hp tg	sigm	2	60,000
L6	50	norm	70	hp tg	lin	2	10
L7	10	standard	80	sigm	sigm	2	10
L8	30	standard	60	sigm	lin	2	10,000
L9	50	standard	70	sigm	hp tg	2	60,000
L10	10	standard	70	hp tg	lin	1	60,000
L11	30	standard	80	hp tg	hp tg	1	10
L12	50	standard	60	hp tg	sigm	1	10,000

Therefore, a total of twelve experiments have been carried out.

3.5. Step 5. Conduct the Experiments

By the OA L_{12}, each experiment has been conducted two times (24 runs in total), corresponding to 12 MLP models whose R^2, *RMSE*, and *MAE* in the training and test set have been calculated in each run. Table 4 shows the measured values of response variable R^2, the mean of R^2 in each run, and the *S/N* ratio obtained from each trial's different networks.

Table 4. R^2, mean, and signal-to-noise (*S/N*) ratio in each run of OA $L_{12.}$

OA	$R^2{}_1$	$R^2{}_2$	Mean	*S/N* Ratio
L1	0.123	0.347	0.235	−15.705
L2	0.914	0.919	0.916	−0.757
L3	0.914	0.845	0.879	−1.135
L4	0.935	0.803	0.869	−1.295
L5	0.524	0.709	0.616	−4.496
L6	0.793	0.91	0.851	−1.458
L7	0.002	0.399	0.200	−50.969
L8	0.955	0.91	0.932	−0.615
L9	0.912	0.899	0.905	−0.863
L10	0.584	0.186	0.385	−12.019
L11	0.903	0.949	0.926	−0.676
L12	0.86	0.549	0.704	−3.683

3.6. Step 6. Predict the Optimum MLP Model and Its Performance

Thus, according to the Taguchi approach, two objective functions are optimized by larger-the-better criterium (§ 2.2.1): the mean of R^2 calculated in each run and the signal/noise ratio (*S/N* Ratio). A standard approach for multi-response optimization is to identify one of the response variables as primary considering it as an objective function and other responses as constraints [54]. Several methods of multiple response optimization have been proposed in the literature. Among them, the utilization of the desirability function is the most efficient approach [55]. In the desirability function analysis (DFA), each response is transformed into a desirability value d_i, and the total desirability function D, which is the geometric mean of the single d_i, is optimized [56]. Desirability D is an objective function that ranges from 0 to 1. If the response is on target, the desirability value will be equal to 1, and DFA will not maximize the desirability value. When the response falls within the tolerance range but not on the desired value, the corresponding desirability will be between 0 and 1 [57]. As the response approaches the target, the desirability value will become closer to 1.

According to the desirability function analysis, the optimal combination of MLP network parameters has been obtained (Figure 5).

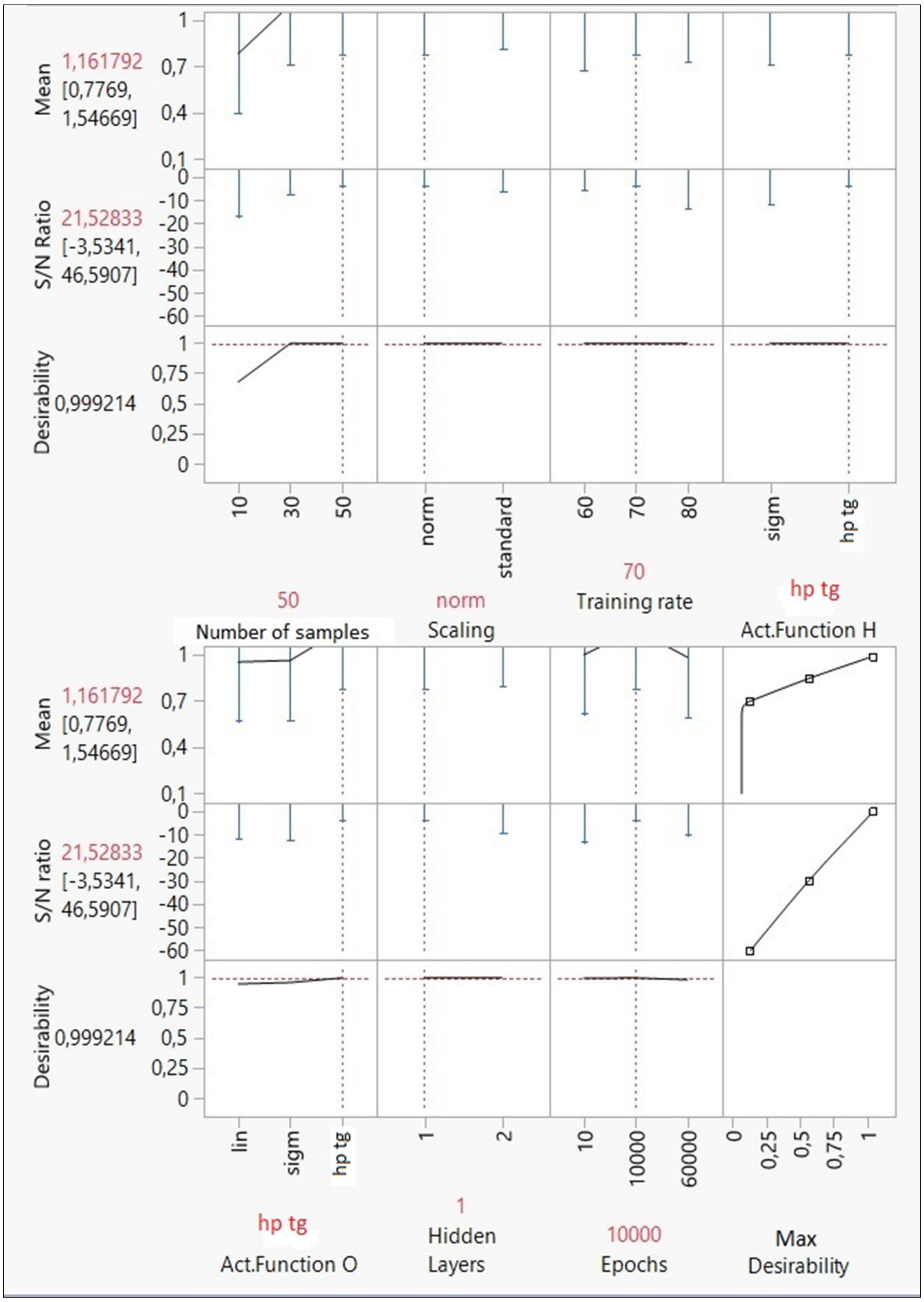

Figure 5. The optimization results according to the desirability function analysis (DFA).

Optimal response and factor values are displayed in red. The desirability function is very close to 1 (0.999214), and the tolerance interval for the *S/N* ratio and mean is

[−3.54, 46.6] and [0.78, 1.55], respectively. The best architecture's features set to maximize the predictive capability of the MLP model is shown in the following outline (Table 5).

Table 5. Optimal architecture's parametric set.

Number of Samples	Input Scaling	Training Rate (%)	Act. Function H	Act. Function O	n. Hidden Layers	Epochs
50	norm	70	hp tg	hp tg	1	10,000

Taguchi results are analyzed in detail below:

- Number of samples: at least 50 samples are required to obtain an optimal model. Thus, a small number of units could cause unbiased predictions.
- Input scaling rule: the normalization of input variables produces better results than the standardization rule.
- Training rate (%): in an optimal MLP model, the training set must consider 70% of database units. Thus, the test set represents the remaining 30%.
- Activation function of hidden and output nodes: the best activation function is the hyperbolic tangent for both hidden and output nodes.
- Number of hidden layers: according to Taguchi's design, a deep network is not the best solution for this analysis; one hidden layer has been more than enough to optimize the forecasts.
- Epochs: the model accuracy has been determined in 10,000 epochs.

3.7. Step 7. Conduct the Verification Experiment

A new experiment (optimal network) has been carried out to compare its results with those obtained from the first MLP model (default network) (§ 2.2). In addition, to make a homogeneous comparison, a new default model has been built using the same 50 samples of the optimal net. Both new models have been an MLP feed-forward network based on 10 inputs, three hidden nodes in one layer, and one output. The architecture's parametric set and performance indicators of three networks (MLP_{def1}, MLP_{def2}, and MLP_{opt}) are summarized as follows (Table 6).

Table 6. Comparison of model's characteristics.

Network Features	$\mathbf{MLP_{def1}}$	$\mathbf{MLP_{def2}}$	$\mathbf{MLP_{opt}}$
Number of samples	64	50	50
Scaling	standard	standard	norm
Training rate	70	70	70
Act. FunctionH	tg hp	tg hp	tg hp
Act. FunctionO	lin	lin	tg hp
Hidden layer	1	1	1
Epochs	10	10	10,000
Rsquare	0.87	0.63	0.93
RMSEtraining	1.41	1.683	0.129
RMSEtest	1.788	30.319	0.025
MAEtraining	0.064	0.102	0.025
MAEtest	0.65	0.488	0.691

By analyzing the MLP_{def1} and MLP_{def2}, the best solution is to consider a higher number of samples increasing performance in case of an arbitrary choice of architecture parametric set. However, the optimal model MLP_{opt}, in which the original dataset has been normalized, has produced, with a longer training time, more reliable predictions than MLP_{def1}: the *RMSE* and *MAE* values (in training and test sets) are lower, and R^2 is 0.93.

Therefore, through the Taguchi approach, it has been possible to find the best design to improve an MLP model performance.

In environmental analysis, relationships between organic and inorganic micro-pollutants are connected to the sampling site's geochemical and lithological properties. Thus, this optimal MLP model is site-specific and the parametric set determined by the Taguchi method is not valid in different locations. For this reason, in various polluted areas, it is necessary to create a new application of the approach to evaluate a best-performing network.

4. Conclusions

The selection of the parametric set of a neural network model is a very challenging issue. A random choice of an MLP's design could compromise the reliability of the network. In this paper, Taguchi's approach for the optimal design of shallow neural networks has been presented, considering an application of the ANNs algorithm in an environmental field, to characterize a polluted soil. Different turning experiments were conducted considering the various combinations of seven architecture parameters (number of samples, scaling, training rate, the activation function of hidden and output nodes, number of hidden layers, and epochs). The optimum levels of parameters have been identified by the desirability function analysis (DFA).

The model so built is another shallow network that is able to produce more reliable predictions through a smaller dataset than the original and a longer training time. The original dataset has been preferred to be normalized than standardized. A hyperbolic tangent has proved to be the best form of the activation function for both hidden layers and output units.

Several benefits can arise from using this method to optimize an ANN's architecture. Firstly, this methodology is the only known method for neural network design that explicitly considers robustness as a significant design criterion. This capability will improve the quality of the neural network designed. Secondly, using the methodology, several parameters of a neural network can be considered simultaneously in the optimization process, evaluating the impact of these factors of interest concurrently. Finally, the Taguchi method is not strictly confined to the design of back-propagation neural networks. Thus, this methodology will allow the rapid development of the best neural network to suit a particular application.

In environmental analysis, this method cannot be generalized since the relations between input and output variables analyzed by the network are affected by various exogenous factors that are difficult to control. Therefore, every analysis requires a new application of the Taguchi method to determine a more performing model. Then, the optimal model can be used for further analysis.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Gardner, M.W.; Dorling, S.R. Artificial neural networks (the multi-layer perceptron)—A review of applications in the atmospheric sciences. *Atmos. Environ.* **1998**, *32*, 2627–2636. [CrossRef]
2. Gomes-Ramos, E.; Venegas-Martinez, F. A Review of Artificial Neural Networks: How Well Do They Perform in Forecasting Time Series? *Analìtika* **2013**, *6*, 7–15.

3. Haykin, S. *Neural Networks a Comprehensive Foundation*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1994.
4. Bishop, C.M. *Neural Networks for Pattern Recognition*, 3rd ed.; Oxford University Press: Oxford, UK, 1995.
5. Mc Culloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biol.* **1943**, *5*, 115–133.
6. Agatonovic-Kustrin, S.; Beresford, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J. Pharm. Biomed. Anal.* **2000**, *22*, 717–727. [CrossRef]
7. Karlik, B.; Vehbi Olgac, A.V. Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. *Int. J. Artif. Intell. Expert Syst.* **2011**, *1*, 111–122.
8. Maier, H.R.; Dandy, G.C. The effect of internal parameters and geometry on the performance of back-propagation neural networks: An empirical study. *Environ. Model. Softw.* **1998**, *13*, 193–209. [CrossRef]
9. Maier, H.R.; Dandy, G.C. Understanding the behaviour and optimising the performance of back-propagation neural networks: An empirical study. *Environ. Model. Softw.* **1998**, *13*, 179–191. [CrossRef]
10. Ross, J.P. *Taguchi Techniques for Quality Engineering*; McGraw-Hill: New York, NY, USA, 1996.
11. Wang, S.H.; Wu, X.; Zhang, Y.D.; Tang, C.; Zhang, X. Diagnosis of COVID-19 by Wavelet Renyi Entropy and Three-Segment Biogeography-Based Optimization. *Int. J. Comput. Int. Syst.* **2020**, *13*, 1332–1344. [CrossRef]
12. Zhang, Y.D.; Sui, Y.; Sun, J.; Zhao, G.; Qian, P. Cat Swarm Optimization applied to alcohol use disorder identification. *Multimed. Tools Appl.* **2018**, *77*, 22875–22896. [CrossRef]
13. Bonfitto, A.; Feraco, S.; Tonoli, A.; Amati, N.; Monti, F. Estimation accuracy and computational cost analysis of artificial neural networks for state of charge estimation in lithium batteries. *Batteries* **2019**, *5*, 47. [CrossRef]
14. Dinesh Kumar, D.; Gupta, A.K.; Chandna, P.; Pal, M. Optimization of neural network parameters using Grey–Taguchi methodology for manufacturing process applications. *J. Mech. Eng. Sci.* **2015**, *229*, 2651–2664. [CrossRef]
15. Ma, L.; Khorasani, K. A new strategy for adaptively constructing multi-layer feed-forward neural networks. *Neurocomputing* **2003**, *51*, 361–385. [CrossRef]
16. Kwon, Y.K.; Moon, B.R. A hybrid neurogenetic approach for stock forecasting. *IEEE Trans. Neural Netw.* **2007**, *18*, 851–864. [CrossRef]
17. Bebis, G.; Georgiopoulos, M.; Kasparis, T. Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing* **1997**, *17*, 167–194. [CrossRef]
18. Benardos, P.G.; Vosniakos, G.C. Optimizing feed-forward artificial neural network architecture. *Eng. Appl. Artif. Intell.* **2007**, *20*, 365–382. [CrossRef]
19. Marijke, K. Genetic Algorithms, Anoverview. 2002. Available online: http://www.meteck.org/gaover.html (accessed on 31 March 2021).
20. Arifovic, J.; Gencay, R. Using genetic algorithms to select architecture of a feed-forward artificial neural network. *Physica* **2001**, *289*, 574–594. [CrossRef]
21. Yang, J.L.; Chen, J.C. A Systematic Approach for Identifying Optimum Surface Roughness Performance in End-Milling Operations. *J. Ind. Technol.* **2001**, *17*, 2–8.
22. Lin, C.; Li, Y.-C.; Lin, H.-Y. Using Convolutional Neural Networks Based on a Taguchi Method for Face Gender Recognition. *Electronics* **2020**, *9*, 1227. [CrossRef]
23. Tabassum, M.; Mathew, K. A Genetic Algorithm Analysis towards Optimization solutions. *Int. J. Digit. Inf. Wirel. Commun.* **2014**, *4*, 124–142. [CrossRef]
24. Larochelle, H.; Jerome, Y.B.; Lamblin, L.P. Exploring Strategies for Training Deep Neural Networks. *J. Mach. Learn. Res.* **2009**, *1*, 1–40.
25. Li, L. Introduction to Multilayer Neural Networks with TensorFlow's Keras. API. Available online: https://towardsdatascience.com/ (accessed on 31 March 2021).
26. Hornik, K. Approximation capabilities of multi-layer feed-forward networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
27. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
28. Sutton, R.S. Two problems with back-propagation and other steepest-descent learning procedures for networks. In Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, MA, USA, 15–17 August 1986; pp. 823–831.
29. Ghaffari, A.; Abdollahi, H.; Khoshayand, M.R.; Soltani Bozchalooi, I.; Dadgar, A.; Rafiee-Tehrani, M. Performance comparison of neural network training algorithms in modeling of bimodal drug delivery. *Int. J. Pharm.* **2006**, *11*, 126–138. [CrossRef] [PubMed]
30. Erb, R.J. Introduction to back-propagation neural network computation. *Pharm. Res.* **1993**, *10*, 165–170. [CrossRef] [PubMed]
31. Leitch, G.; Tanner, J.E. Economic Forecast Evaluation: Prots Versus the Conventional Error Measures. *Am. Econ. Rev.* **1991**, *81*, 580–590.
32. Kuroe, Y.; Yoshid, M.; Mori, T. On Activation Functions for Complex-Valued Neural Networks. In *ICANN/ICONIP 2003 Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2714.
33. Shmueli, G.; Patel, N.R.; Bruce, P.C. *Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro*; John Wiley & Sons Inc.: Hoboken, NJ, USA, 2016.
34. Musumeci, F.; Rottondi, C.; Nag, A.; Macaluso, I.; Zibar, D.; Ruffini, M.; Tornatore, M. Survey on application of machine learning techniques in optical networks. *IEEE Commun. Surv. Tutor.* **2018**, 1–27.
35. Demuth, H.; Beale, M.; Hagan, M. 2008; Neural Network ToolboxTM6. *User's Guide MathWorks* **2008**, *9*, 259–265.
36. Shahin, M.A.; Maier, H.R.; Jaksa, M.B. Data division for developing neural networks applied to geotechnical engineering. *J. Comput. Civ. Eng. ASCE* **2004**, *18*, 105–114. [CrossRef]

37. Gallo, C. Artificial Neural Networks Tutorial. In *Enciclopedia of Information, Science and Technology*, 3rd ed.; MEHDI Khosrow-Pour Information Resources Management Association: Washington, DC, USA, 2015; pp. 179–189.
38. Kotsiantis, S.B.; Kanellopoulos, D.; Pintelas, P.E. Data Pre-processing for Supervised Learning. *CompSci* **2006**, *1*, 1306–4428.
39. Shanker, M.; Hu, M.Y.; Hung, M.S. Effect of data standardization on neural network training. *Omega* **1996**, *24*, 385–397. [CrossRef]
40. Mjalli, F.S.; Al-Asheh, S.; Alfadala, H.E. Use of artificial neural network black-box modeling for the prediction of wastewater treatment plants performance. *J. Environ. Manag.* **2007**, *83*, 329–338. [CrossRef]
41. Hush, D.; Horne, B.G. Progress in supervised neural networks. *IEEE Signal Process.* **1993**, *10*, 8–39. [CrossRef]
42. Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
43. Twomey, J.M.; Smith, A.E. Performance Measures, Consistency and Power for Artificial Neural Network Models. *Mathl. Comput. Model.* **1995**, *21*, 243–258. [CrossRef]
44. Afan, H.; El-Shafie, A.; Yaseen, Z.; Hameed, M.; Wan Mohtar, H.; Hussain, A. ANN Based Sediment Prediction Model Utilizing Different Input Scenarios. *Water Resour. Manag.* **2015**, 29. [CrossRef]
45. Chaloulakou, A.; Saisana, M.; Spyrellis, N. Comparative assessment of neural networks and regression models for forecasting summertime ozone in Athens. *Sci. Total Environ.* **2003**, *313*, 1–13. [CrossRef]
46. Nalbant, M.; Gökkaya, H.; Sur, G. Application of Taguchi method in the optimization of cutting parameters for surface roughness in turning. *Mater. Des.* **2007**, *28*, 1379–1385. [CrossRef]
47. Antony, J. *Fractional Factorial Designs in Design of Experiments for Engineers and Scientists*, 2nd ed.; Elsevier Insights: London, UK, 2014; pp. 87–112.
48. Bonelli, M.G.; Ferrini, M.; Manni, A. Artificial neural networks to evaluate organic and inorganic contamination in agricultural soils. *Chemosphere* **2017**, *186*, 124–131. [CrossRef]
49. IBM SPSS Neural Networks 25. Available online: https://www.ibm.com/downloads/cas/JPKAPO4L (accessed on 31 March 2021).
50. Charan Kumar, G.; Varalakshmi, M.; Ankita, T.; Rajyalakshmi, K. Modified Taguchi Approach for optimizing the process parameter using the fictitious parameter. *J. Phys.Conf. Ser.* **2019**, *1344*, 1–11. [CrossRef]
51. Fratilia, D.; Caizar, C. Application of Taguchi method to selection of optimal lubrication and cutting conditions in face milling of AlMg3. *J. Clean. Prod.* **2011**, *19*, 640–645. [CrossRef]
52. Sharma, N.; Ragsdell, K. *Quality Loss Function—Common Methodology for Nominal-The-Best, Smaller-The-Better, and Larger-The-Better Cases*; SAE Technical Papers; SAE International: Warrendale, PA, USA, 2007. [CrossRef]
53. Antony, J. Multi-response optimization in industrial experiments using Taguchi's quality loss function and principal component analysis. *Qual. Reliab. Engng. Int.* **2000**, *16*, 3–8. [CrossRef]
54. Phadke, M.S. *Quality Engineering Using Robust Design*; Prentice Hall: Englewood Cliffs, NJ, USA, 1989.
55. John, B. Application of desirability function for optimizing the performance characteristics of carbonitrided bushes. *Int. J. Ind. Eng. Comput.* **2013**, *4*, 305–314. [CrossRef]
56. Derringer, G. A balancing act: Optimising product's properties. *Qual. Prog.* **1994**, *27*, 51–58.
57. Subrahmanyam, A.; Maheswara Rao, C.; Naga Raju, B. Taguchi based desirability function analysis for the optimization of multiple performance characteristics. *IJMTER* **2018**, *5*, 168–175.