applied sciences

Article

A Novel Inverse Kinematic Solution of a Six-DOF Robot Using Neural Networks Based on the Taguchi Optimization Technique

Teodoro Ibarra-Pérez [1,*], José Manuel Ortiz-Rodríguez [2], Fernando Olivera-Domingo [1], Héctor A. Guerrero-Osuna [3], Hamurabi Gamboa-Rosales [3] and Ma. del Rosario Martínez-Blanco [2,*]

1 Instituto Politécnico Nacional, Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas (UPIIZ), Zacatecas 98160, Mexico
2 Laboratorio de Innovación y Desarrollo Tecnológico en Inteligencia Artificial (LIDTIA), Universidad Autónoma de Zacatecas, Zacatecas 98000, Mexico
3 Unidad Académica de Ingeniería Eléctrica, Universidad Autónoma de Zacatecas, Zacatecas 98000, Mexico
* Correspondence: tibarrap@ipn.mx (T.I.-P.); mrosariomb@uaz.edu.mx (M.d.R.M.-B)

Abstract: The choice of structural parameters in the design of artificial neural networks is generally based on trial-and-error procedures. They are regularly estimated based on the previous experience of the researcher, investing large amounts of time and processing resources during network training, which are usually limited and do not guarantee the optimal selection of parameters. This paper presents a procedure for the optimization of the training dataset and the optimization of the structural parameters of a neural network through the application of a robust neural network design methodology based on the design philosophy proposed by Genichi Taguchi, applied to the solution of inverse kinematics in an open source, six-degrees-of-freedom robotic manipulator. The results obtained during the optimization process of the structural parameters of the network show an improvement in the accuracy of the results, reaching a high prediction percentage and maintaining a margin of error of less than 5%.

Keywords: backpropagation; optimization methods; inverse kinematics; robotics

check for updates

Citation: Ibarra-Pérez, T.; Ortiz-Rodríguez, J.M.; Olivera-Domingo, F.; Guerrero-Osuna, H.A.; Gamboa-Rosales, H.; Martínez-Blanco, M.d.R. A Novel Inverse Kinematic Solution of a Six-DOF Robot Using Neural Networks Based on the Taguchi Optimization Technique. *Appl. Sci.* **2022**, *12*, 9512. https://doi.org/10.3390/app12199512

Academic Editor: Luis Gracia

Received: 24 August 2022
Accepted: 17 September 2022
Published: 22 September 2022

1. Introduction

One of the main problems in the design of neural networks is the selection of the structural parameters of the network and their corresponding values before performing the training. In this work, the robust design artificial neural network (RDANN) methodology is used. The main focus of this methodology is based on reducing the number of experiments that can be carried out using the factorial fractional method, a statistical procedure based on the robust design philosophy proposed by Genichi Taguchi. This technique allows one to set the optimal settings on the control factors to make the process insensitive to noise factors [1,2].

Currently, the selection of the structural parameters in the design of artificial neural networks (ANNs) remains a complex task. The design of neural networks implies the optimal selection of a set of structural parameters in order to obtain greater convergence during the training process and high precision in the results. In [1], the feasibility of this type of approach for the optimization of structural parameters in the design of a backpropagation artificial neural network (BPANN) for the determination of operational policies in a manufacturing system is demonstrated, where it is shown that the Taguchi method allows designers to improve the performance in the learning speed of the network and the precision in the obtained results.

Most designers select an architecture type and determine the various structural parameters of the chosen network. However, there are no clear rules on how to choose those parameters in the selected network architecture, although these parameters determine the success of the network training. The selection of the structural parameters of the network is generally carried out through the implementation of conventional procedures based on trial and error, as shown in Figure 1, where a significant number of ANN models are generally implemented in comparison with other unconventional procedures [3–6].

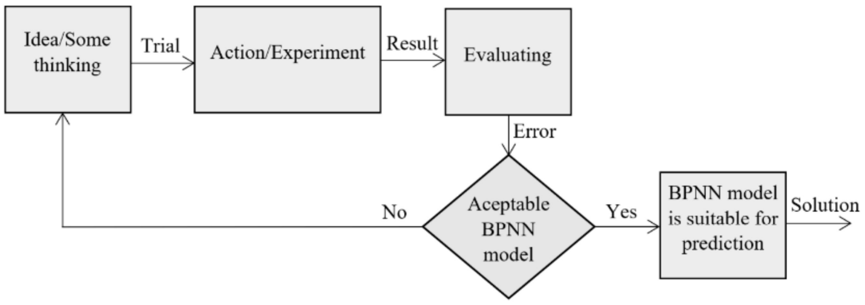


Figure 1. Trial-and-error procedure.

In this case, if a desired level of performance is not maintained, the levels in the previously established design parameters are changed until the desired performance is obtained. In each experiment, the responses are observed in order to determine the appropriate levels in the design of the structural parameters of the network [7].

A drawback in the use of this type of procedure is that one parameter is evaluated, while the others are kept at a single level, so the level selected in a variable may not necessarily be the best at the end of the experiment, since it is very likely that most of the layout variables involved will change their value. A possible solution could be that all the possible combinations in the parameters are evaluated, that is, to carry out a complete factorial design. However, the number of combinations can be very large due to the number of levels and previously established design parameters, so this method could be computationally expensive and time-consuming.

Due to all these limitations, the scientific community has shown special interest in the implementation of new approaches and procedures applied to the optimization of structural parameters in the search to generate better performance in ANNs [8–13].

Currently, ANNs can be trained to solve problems that can be complex for a human or a conventional computer, since they allow obtaining results with a high degree of precision and a significant reduction in error in real-time applications. In recent decades, the use of ANNs has been successfully applied in different fields, including pattern recognition, classification, identification, voice, vision, control systems, and robotics, the latter of which has raised special interest among researchers in the field, particularly the solution of the inverse kinematics in manipulators with six or more degrees of freedom, due to the great flexibility of control that they present for the execution of very complex tasks [14–17].

In [18], a BPNN algorithm is proposed, optimized by Fruit Fly Optimization Algorithm (FOA), to find the solution of the inverse kinematics in a four-DOF robot, obtaining an output error range -0.04686–0.1271 smaller than that obtained by a BPNN. In [19], a BPNN algorithm, optimized by means of particle swarm optimization (PSO), is studied to solve the inverse kinematic problem in a six-DOF UR3 robot applied in puncture surgery, where convergence in the precision of the results, as well as the speed and generalization capacity of the proposed network, is improved. In [20], a deep learning approach is proposed to solve the inverse kinematics in a seven-DOF manipulator. The approach used allows it to be fast, easy to implement, and more stable, allowing less sensitivity in hyperparameters. In [21], a combination of swarm intelligence (SI) and the product of exponentials (PoEs) is used to solve the inverse kinematics in a seven-DOF manipulator, where they are compared with the conventional inverse kinematics and standard PSO algorithms. In [22], the main approach is based on a redundant manipulator inverse kinematic problem that is formulated as a quadratic programming optimization problem solved by different types of recurrent neural networks. In [23], an approach is proposed to address the complexity of solving the inverse kinematics in a seven-DOF serial manipulator through an algorithm based on the Artificial Bee Colony (ABC) optimization algorithm, where two control parameters are used in order to adjust the search to optimize the distribution of the sources. In [24], an optimization approach is shown in the planning of the trajectories applied in a five-bar parallel robot for real-time control, minimizing the trajectory time and avoiding singularities in the parallel manipulator, achieving an error of less than 0.7° at the joints.

Factorial experimental design is a statistical technique used to identify and measure the effect that one variable has on another variable of interest. In 1920, R. A. Fisher studied multiple factors in the agricultural field to determine the effect of each factor on the response variable, as well as the effect of the interactions between factors on this variable. This method is known as the factorial design of experiments. Factors are variables that determine the functionality of a product or process and significantly influence system performance and can usually be controlled. To evaluate the impact of each variable, the factors must establish at least two levels; therefore, given k factors with l levels, a complete factorial design that includes all the possible combinations between these factors and levels will produce a total of l^k experimental runs. Obviously, as k or l increases, the number of experiments may become unfeasible to carry out, since a significant number of factors would imply a large number of experiments. For this reason, fractional factorial designs have been introduced, which require only a fraction of a run, unlike a complete factorial design, and which allow estimating a sufficient number of effects [25,26].

Genichi Taguchi is considered to be the author of robust parameter design through a procedure focused on reducing variation and/or sensitivity to noise in the design of products or processes, which is based on the concept of fractional factorial design. Through the implementation of orthogonal arrays (OA) and fractional factorial design, it is possible to analyze a wide range of parameters through a reduced number of experiments, ensuring a balanced comparison between the factors involved and the interaction with their different levels [2,27,28].

The Taguchi method is applied in four stages:

1. *Selection of design and noise variables*. In this stage, the most important parameters for the product/process are considered, taking into account the quality characteristics. Generally, there are variables that can be controlled by the user and others that cannot. These types of variables are known as design and noise factors, respectively, which have an important influence on the operation of the product/process. They can be determined mainly by answering the following questions: What is the optimal design condition? What factors contribute to the results and to what extent? What will be the expected result?
2. *Design and experimentation*. An OA is established, which contains the organization of the experiment taking into account the levels established for each of the factors in order to minimize the effects produced by noise factors. In other words, the adjustments made to the factors must be determined in such a way that there is the least variation in the response of the product/process, and the mean is established as close as possible to the desired objective. The OA allows the implementation of a balanced design in the weighting of the pre-established levels for each factor involved since it is possible to evaluate various factors with a minimum number of tests, obtaining a considerable amount of information through the application of few tests. The mean and variance of the response obtained in the OA configuration are combined into a single performance measure known as the signal-to-noise ratio (S/N).
3. *Analysis of results*. The S/N ratio is a quality indicator by which the effect produced on a particular parameter can be evaluated. The variation in the response obtained in dynamic characteristics, the S/N ratio, is shown below in the following equation:

$$S/N = 10 \cdot log_{10}\left(\frac{\beta_i}{MSE_i}\right), \tag{1}$$

where β_i is the square of the largest value of the signal, and MSE_i represents the root mean square deviation in the performance of the neural network, or in other words, the mean square of the distance between the measured response and the best fit line.

A valid robustness measure is related to obtaining the highest values in the S/N ratio, because the configurations of control factors that minimize the effects on noise factors can be identified.

4. *Execution and confirmation of tests in optimal conditions.* In this stage, a confirmation experiment is carried out by performing training with optimal design conditions in order to calculate the performance robustness measure and verify if this value is close to the predicted value.

Inverse Kinematics with ANNs

During the last decade, robotics had an outstanding development in the industry, particularly in aerospace, military, and medical areas, among others, especially in manipulators with a large number of degrees of freedom (DOF), due to their high flexibility and control to perform complex tasks [17,29,30].

Modern manipulators, usually kinematically redundant, allow complex tasks to be solved with high precision in the results. These types of manipulators have at least six DOF, allowing greater flexibility and mobility to perform complex tasks. The complexity in manipulator control design based on an inverse kinematic solution approach can be computationally complex, due to the nonlinear differential equation systems that are usually present. Traditional methods with geometric, iterative, and algebraic approaches have certain disadvantages and can often be generically inappropriate or computationally expensive [16,31].

The ANNs present major advantages related to nonlinearity, parallel distribution, high learning capacity, and great generalization capacity, and they can maintain a high calculation speed, thus fulfilling the real-time control requirements. Consequently, various approaches have been proposed by the scientific community in the use of intelligent algorithms applied to the control of robotic manipulators such as the use of ANNs [19,20,32,33], genetic algorithms [31,33–38], recurrent neural networks (RNNs) [37], [38], optimization algorithms [18,23,39,40], and the use of neural networks and optimization methods for parallel robots [24,41].

The organization of this work is as follows: In Section 2.1, the kinematic model of the Quetzal manipulator is established. Section 2.2 describes the procedure for generating the training and testing dataset. Section 2.3 describes the implementation of the RDANN methodology for the optimization of structural parameters in the BPNN. In Section 3, the results obtained are subjected to a reliability test stage through the use of a cross-validation method to verify that the dataset is statistically consistent. The results of training in the optimized BPNN show a significant improvement in the accuracy of the results obtained compared with the use of conventional procedures based on trial-and-error tests.

2. Materials and Methods

In this paper, a robust design model is presented through a methodological and systematic approach based on the design philosophy proposed by Genichi Taguchi. The integration of optimization processes and ANN design are methodological tools that allow the performance and generalization capacity in ANN models to be improved. In this study, an RDANN methodology was used, which was initially proposed for the reconstruction of spectra in the field of neutron dosimetry by means of ANNs [7].

The RDANN methodology was used to estimate the optimal structural parameters in a BPNN to calculate the inverse kinematics in a six-DOF robot, where the main objective was the development of accurate and robust ANN models. In other words, it was sought that the selection of the structural parameters of the proposed model allows us to obtain the best possible performance in the network.

2.1. Kinematic Analysis

The robot called Quetzal is based on an open source, 3D-printable, and low-cost manipulator [42]. The modeling and representation were carried out using the Denavit–

Hartemberg (D–H) parameters to obtain a kinematic model through four basic transformations that are determined based on the geometric characteristics of the Quetzal manipulator to be analyzed [43]. The D–H parameters are shown in Table 1.

Table 1. D–H parameters of the Quetzal manipulator.

i	**Link Offset d_i (cm)**	**Joint Angle θ_i (rad)**	**Link Length a_{i-1} (cm)**	**Twist Angle α_{i-1} (rad)**
1	20.2	θ_1	0	$\pi/2$
2	0	θ_2	16	0
3	0	$\theta_3 + \pi/2$	0	$\pi/2$
4	19.5	θ_4	0	$-\pi/2$
5	0	θ_5	0	$\pi/2$
6	6.715	θ_6	0	0

The basic transformations represent a sequence of rotations and translations, where the reference system of element i is related to the system of element $i-1$. The transformation matrix is given by Equation (2).

$$^{i-1}_{i}T = R_Z(\theta_i)D_Z(d_i)D_{X,}(a_{i-1})R_X(\alpha_{i-1}) \tag{2}$$

Carrying out the multiplication of the four matrices, Equation (3) is obtained:

$$^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_{i-1} & s\theta_i s\alpha_{i-1} & a_{i-1}c\theta_i \\ s\theta_i & c\theta_i c\alpha_{i-1} & -c\theta_i s\alpha_{i-1} & a_{i-1}s_{\theta i} \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where $^{i-1}_{i}T$ is the D–H transformation matrix from coordinate system i to $i-1$. $R_Z(\theta_i)$ is the rotation matrix representing a rotation θi around the Z axis, $D_Z(d_i)$ is the translation matrix representing a translation of d_i on the Z axis, $D_{X,}(a_{i-1})$ is the translation matrix representing a translation of a_{i-1} on the X axis, $R_X(\alpha_{i-1})$ is the rotation matrix representing a rotation α_{i-1} around the X axis, $c\theta_i$ is shorthand for cos $cos(\theta_i)$, $s\theta_i$ is shorthand for $sin(\theta_i)$, etc. The transformation matrices of each of the joints are multiplied to obtain the initial position of the end effector in the base reference system, as shown in Equation (4).

$$^{0}_{6}T = {}^{0}_{1}A \cdot {}^{1}_{2}A \cdot {}^{2}_{3}A \cdot {}^{3}_{4}A \cdot {}^{4}_{5}A \cdot {}^{5}_{6}A = \begin{bmatrix} {}^{0}_{6}R_{3\times3} & {}^{0}_{6}P_{3\times1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Therefore, the equation of the forward kinematics of the Quetzal manipulator can be expressed as shown in Equation (5).

$$F_{forward\_k}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \left(p_x, p_y, p_z, n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\right) \tag{5}$$

As shown in Equation (5), the position of the end effector of the manipulator can be obtained from the angular values of the six joints of the manipulator. However, in practice, it is necessary to obtain the angles at each of the joints through a given position, so it is necessary to calculate the inverse kinematics, which can be expressed as shown in Equation (6).

$$F_{inverse\_k}\left(p_x, p_y, p_z, n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\right) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \tag{6}$$

Solving (4) gives the orientation and position of the final effector in regard to the reference system, as shown in Equation (7), where the position vector $[p] = \{42.215, 0, 20.2\}$ and the orientation vector $[n\ o\ a] = \{0, 0, 1, 0, -1, 0, 1, 0, 0\}$.

$$T_0^6 = \begin{bmatrix} 0 & 0 & 1 & 42.215 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 20.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The graphic representation of the initial position of the Quetzal robotic manipulator is shown in Figure 2 through a simulation carried out with the Robotics Toolbox for MATLAB software [44].

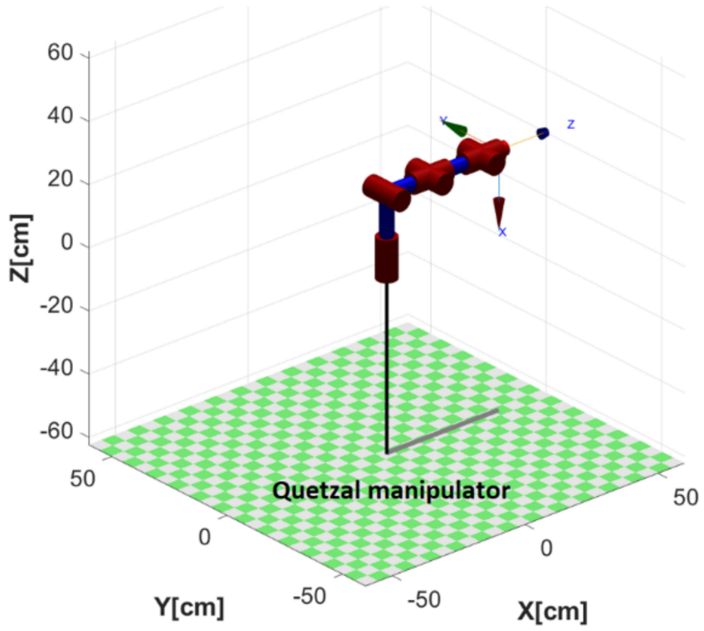


Figure 2. Representation of the initial position of the Quetzal manipulator.

2.2. Training and Testing Datasets

The dataset was generated from the equations obtained in the forward kinematics of the Quetzal manipulator. The variables involved in the proposed dataset were the orientation vector $[n\ o\ a] = \{n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\}$, the position vector $[p] = \{p_x, p_y, p_z\}$, and the vector of joint angles $[\theta] = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$, for a total of 18 variables.

Table 2 shows the ranges of movement established for each of the joints in the workspace of the manipulator. Dataset X was generated with a spatial resolution of $25 \times 25 \times 25 \times 25 \times 25 \times 25 \times 25$ in a six-dimensional matrix with 18 variables involved. The total size of the dataset was 4,394,531,250 data, occupying an approximate physical memory space of 32.74 Gb due to the data class of type $Single - precision$ used in the study, where the data are stored as 4-byte (32-bit) floating point values.

Table 2. Range of motion and step size at each of the joints.

Description	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Min	0	0	2π	0	2π	0
Max	2π	π	$\pi/2$	2π	$\pi/2$	2π
Step	$\pi/12$	$\pi/24$	$\pi/24$	$\pi/12$	$\pi/24$	$\pi/12$
Total steps	25	25	25	25	25	25

Figure 3 shows a graphical representation of the workspace using a 3D data scatter plot corresponding to the position vector $[p]$ based only on the joints $\theta_1, \theta_2, \theta_4, \theta_5$ and θ_6, where it is possible to appreciate the workspace of the robotic manipulator without taking into account joint θ_3. The illustrated workspace was generated from the forward kinematic equations as a function of the six joints.

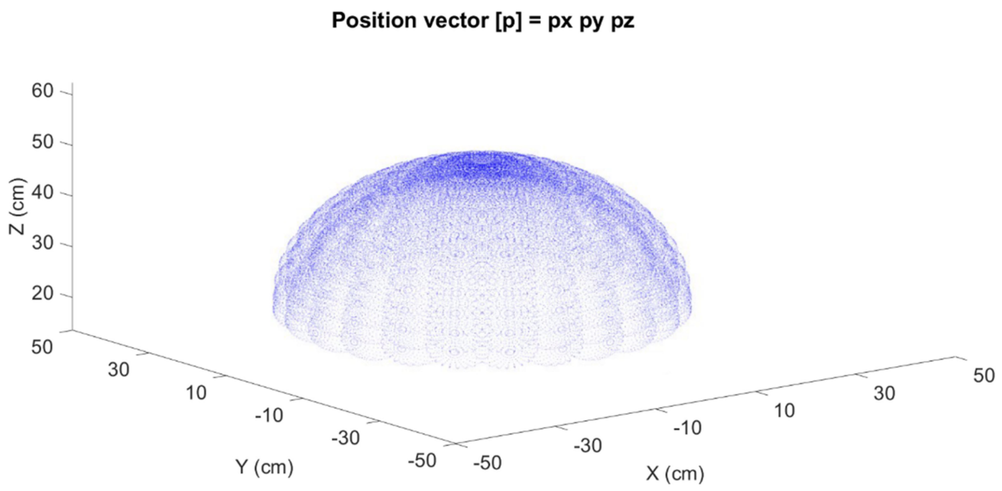


Figure 3. Position vector $[p] = \{p_x, p_y, p_z\}$ in function of joints $\theta_1, \theta_2, \theta_4, \theta_5$ and θ_6.

In order to process the enormous volume of data in a conventional processor, a data reduction filter (DRF) based on linear systematic sampling (LSS) was applied to reduce the set to a size of 190.7 Kb in memory [45]. The data were processed on an 8-core AMD Ryzen 7 5000 series processor with a base clock of 1.8 GHz, 16 GB of RAM, and integrated Radeon 16-thread graphics with a maximum clock of 4.3 GHz [46]. The dataset and code are available in Supplementary Materials.

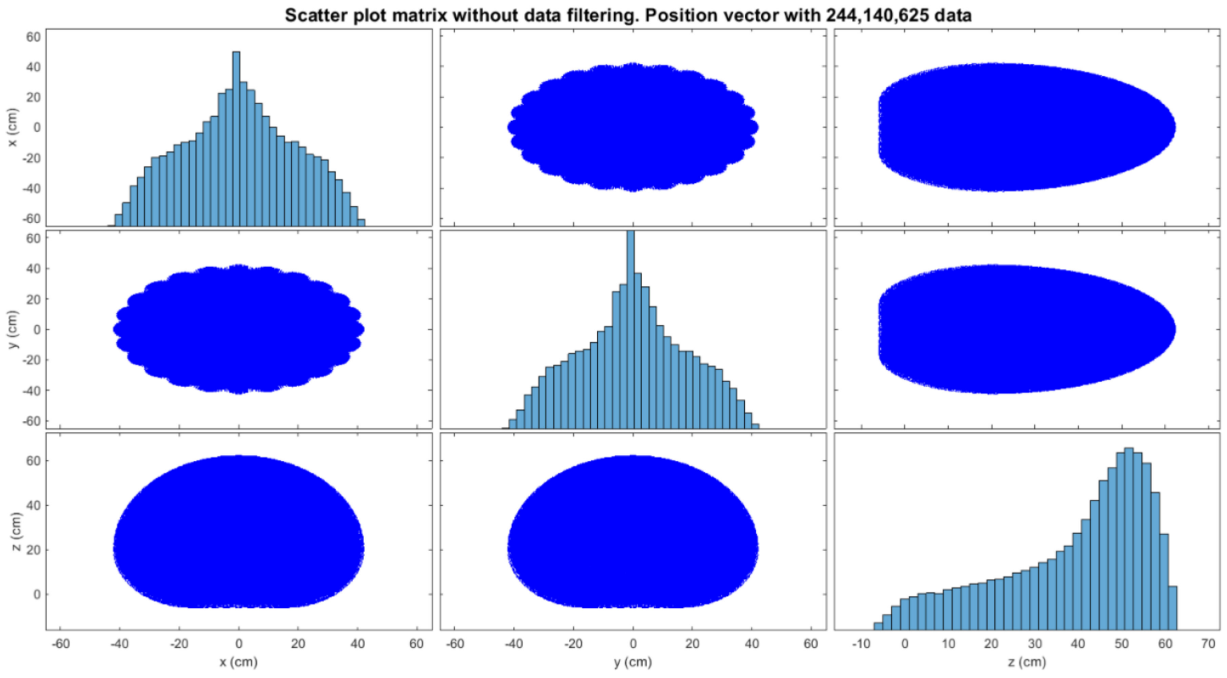
Figure 4 shows the scatter matrices corresponding to the position dataset before and after applying the FRD filter, where a reduction of 99.99% of the data was obtained with a size of 24,410 data. In Figure 4b, it is shown that the data maintain a constant and uniform distribution with respect to the dataset of Figure 4a.

The data were normalized with a mean of zero in the range from -1 to 1 using Equation (8), where *data* is the data to normalize, *min* is the minimum value of the dataset, *range* is a value established by the difference between the maximum value and the minimum value of the dataset, h and l are the maximum and the minimum desired values for normalization.

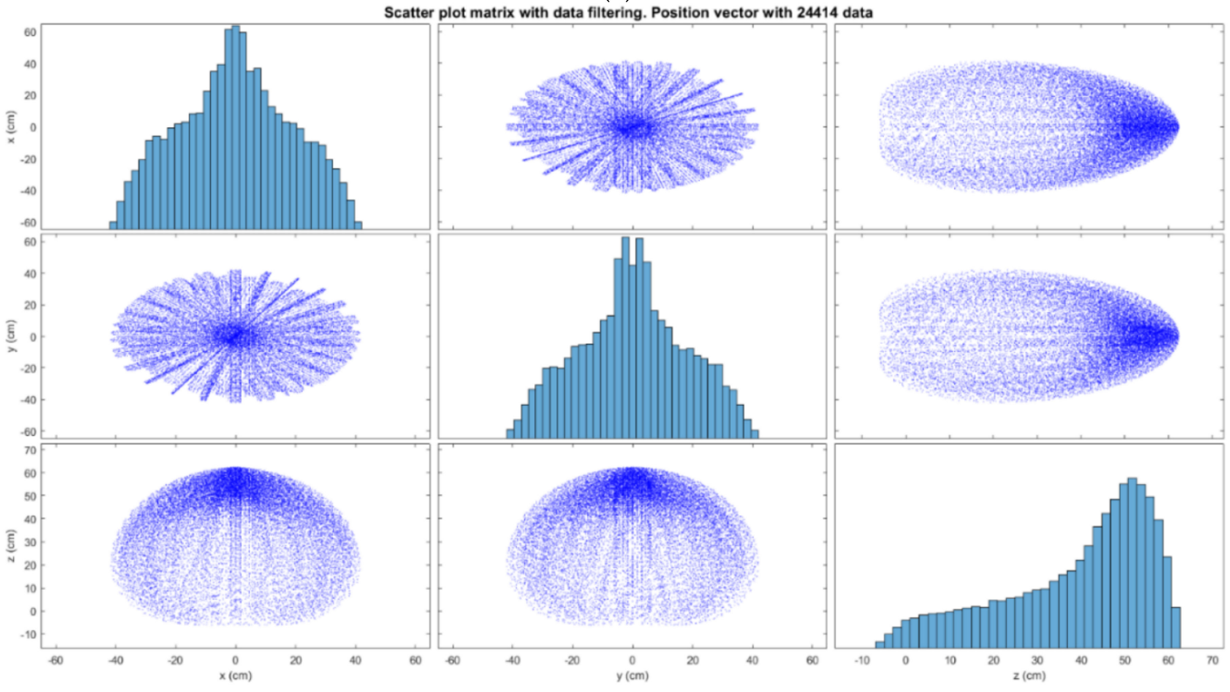
$$DataNorm = \left(\frac{data - min}{range} \times (h - l) \right) + l \tag{8}$$

2.3. Robust Design Methodology

Figure 5 shows the RDANN methodology based on the Taguchi philosophy that consists of four stages. The designer must know the problem and choose an appropriate network model, as well as the parameters involved in the design of the network for its optimization (planning stage). By implementing an OA and systematically training a small number of ANN models (experimentation stage), the response to be analyzed is determined using the S/N relationship of the Taguchi method (analysis stage). Finally, through a confirmation process, the best performance values of the model are obtained (confirmation stage).



(**a**)



(**b**)

Figure 4. Scatter matrix of the position dataset: (**a**) before applying the reduction filter; (**b**) after applying the data reduction filter.

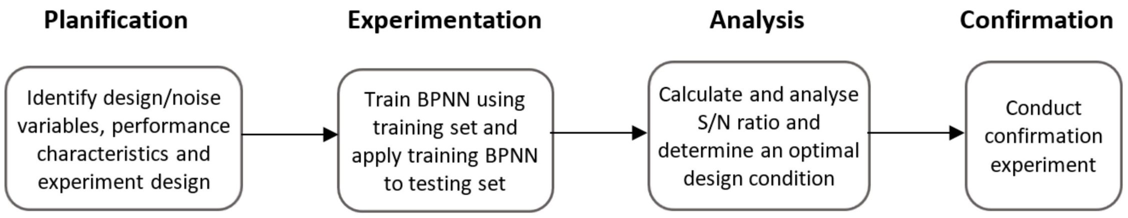

Planification
Experimentation
Analysis
Confirmation
Identify design/noise variables, performance characteristics and experiment design
Train BPNN using training set and apply training BPNN to testing set
Calculate and analyse S/N ratio and determine an optimal design condition
Conduct confirmation experiment

Figure 5. Robust design methodology for the optimization of structural parameters.

The graphic representation of the BPNN used in this work is shown in Figure 6, with 12 input variables and 6 output variables that correspond to the position vector $[p] = \{p_x, p_y, p_z\}$ and orientation vector $[n\ o\ a] = \{n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\}$ as input and the vector of joint angles $[\theta] = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ as output.

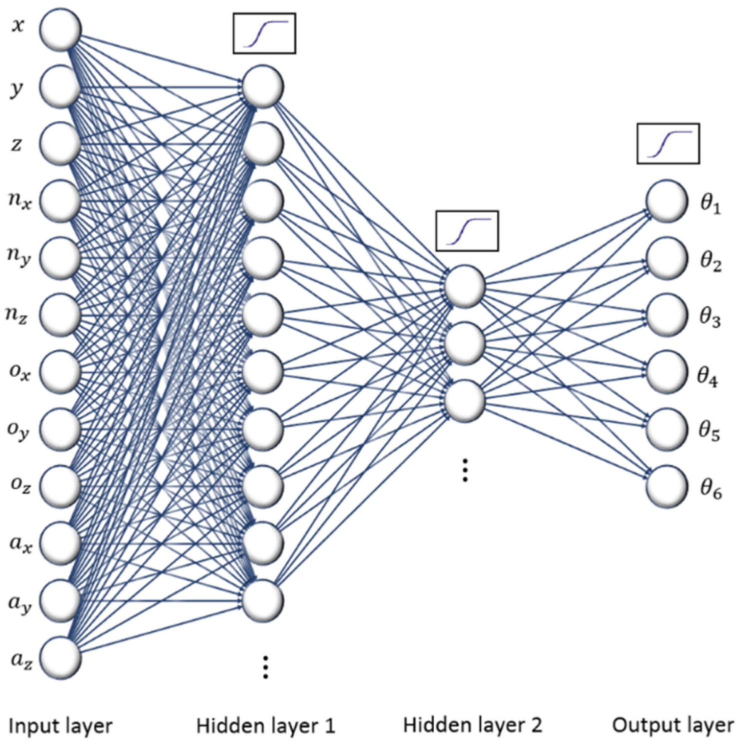


Figure 6. BPNN network topology used in this study.

2.3.1. Planning Stage

In this stage, the design variables, noise, and the objective function are identified. The objective function is defined according to the purpose and requirements of the system. In this work, the objective function is related to the prediction or classification errors between the calculator data and the data predicted by the ANN model during the testing stage. The performance at the output of the ANN or the mean square error (MSE) is used as the objective function and is described in the following equation:

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta_i^{PREDICTED} - \theta_i^{ORIGINAL})^2} \tag{9}$$

In this case, N represents the number of attempts, $\theta_i^{PREDICTED}$ represents the set of joint values that are predicted by the BPANN, and $\theta_i^{ORIGINAL}$ represents the set of joint values.

The design variables correspond to those that can be controlled by the user, such as the number of neurons in the first layer, the number of neurons in the second layer, the momentum constant, and the learning rate. By contrast, the noise variables are commonly not directly controlled by the user in most cases, such as the initialization of synaptic weights that are generally assigned randomly, the size of the training sets versus test sets, and the random selection of training and test sets. According to the requirements of the problem, the user can choose the factors related to variation in the system during the optimization process. Four design variables and three noise variables were selected because they were directly involved with the performance of the ANN, as described below in Table 3 with their respective configuration levels.

In terms of variables, A is the number of neurons in the first hidden layer; B is the number of neurons in the second hidden layer; C is the momentum constant, which allows the stabilization of the updating of each of the synaptic weights taking into account the sign of the gradient; D is the learning rate, which allows us to define the cost that the gradient has in updating a weight because the increase or decrease in the synaptic weight is related to the magnitude of the proposed value, so it may or may not affect the convergence of the MSE, causing instability and divergence [47]. X is the initial set of weights, Y is the size in proportions of the dataset, and Z is the random selection from the training and testing set.

Table 3. Design and noise variables.

Variables	Level 1	Level 2	Level 3
A	L1	L2	L3
B	L1	L2	L3
C	L1	L2	L3
D	L1	L2	L3
X	L1	L2	L3
Y	L1	L2	L3
Z	L1	L2	L3

Once the variables and their respective levels were chosen, a suitable OA was chosen to carry out the training sessions. An OA is described as $L_r(S^c)$, where r represents the number of rows, c represents the number of columns, and s represents the number of levels in each of the columns. In this experiment, the columns of the OA represent the parameters to be optimized, and the rows represent the tests carried out by combining the three proposed levels.

2.3.2. Experimentation Stage

The success in this stage depends on an adequate choice of the OA because, in this process, a series of calculations are carried out in order to evaluate the interaction and the effects produced between the variables involved through a reduced number of experiments. For the implementation of a robust design, Taguchi suggests the use of a configuration in two crossed OAs with $L_9(3^4)$ and $L_4(3^2)$, as shown below in Table 4.

Table 4. Recording of responses in trials using crossed OAs with $L_9(3^4)$ and $L_4(3^2)$ configurations.

Trial No.	A	B	C	D	G1	G2	G3	G4	Average	S/N
1	1	1	1	1	$\text{Resp}_{1,1}$	$\text{Resp}_{2,1}$	$\text{Resp}_{3,1}$	$\text{Resp}_{4,1}$	Avg1	SN1
2	1	2	2	2	$\text{Resp}_{1,2}$	$\text{Resp}_{2,2}$	$\text{Resp}_{3,2}$	$\text{Resp}_{4,2}$	Avg2	SN2
3	1	3	3	3	$\text{Resp}_{1,3}$	$\text{Resp}_{2,3}$	$\text{Resp}_{3,3}$	$\text{Resp}_{4,3}$	Avg3	SN3
4	2	1	2	3	$\text{Resp}_{1,4}$	$\text{Resp}_{2,4}$	$\text{Resp}_{3,4}$	$\text{Resp}_{4,4}$	Avg4	SN4
5	2	2	3	1	$\text{Resp}_{1,5}$	$\text{Resp}_{2,5}$	$\text{Resp}_{3,5}$	$\text{Resp}_{4,5}$	Avg5	SN5
6	2	3	1	2	$\text{Resp}_{1,6}$	$\text{Resp}_{2,6}$	$\text{Resp}_{3,6}$	$\text{Resp}_{4,6}$	Avg6	SN6
7	3	1	3	2	$\text{Resp}_{1,7}$	$\text{Resp}_{2,7}$	$\text{Resp}_{3,7}$	$\text{Resp}_{4,7}$	Avg7	SN7
8	3	2	1	3	$\text{Resp}_{1,8}$	$\text{Resp}_{2,8}$	$\text{Resp}_{3,8}$	$\text{Resp}_{4,8}$	Avg8	SN8
9	3	3	2	1	$\text{Resp}_{1,9}$	$\text{Resp}_{2,9}$	$\text{Resp}_{3,9}$	$\text{Resp}_{4,9}$	Avg9	SN9

2.3.3. Analysis Stage

Through the S/N ratio, a quantitative evaluation is carried out, where the mean and the variation in the responses measured by the ANN with different design parameters are considered. The unit of measure is the decibel, and the formula is described as follows:

$$S/N = 10 \cdot log_{10}(MSD) \tag{10}$$

In this case, MSD is the root mean square deviation in the ANN performance. The best topology is considered when more signals and less noise are obtained; therefore, a high S/N ratio at this stage allows us to identify the best design values in the BPANN with the help of statistical analysis with the JMP software.

2.3.4. Confirmation Stage

In this stage, the value of the robustness measure is obtained based on the specifications and optimal conditions of the design. A confirmation experiment is carried out using the optimal design conditions that were previously chosen, in order to verify if the calculated value is close to the value predicted by the BPANN.

3. Results

In this work, the RDANN methodology was used for the optimal selection of the structural parameters in a feed-forward backpropagation network, known as BPNN, to find the solution to the inverse kinematics in a Quetzal robot. For the BPNN training, the "resilient backpropagation" training algorithm and $mse = 1E^{-4}$ were selected. In accordance with the RDANN methodology, an OA corresponding to the design and noise variables, respectively, was implemented in configurations $L_9(3^4)$ and $L_4(3^2)$ to determine the response to the tests during the 36 training sessions carried out.

The results obtained after applying the RDANN methodology are presented in the next sections.

3.1. Planning Stage

Table 5 shows the design and noise variables with their respective assigned values for the different levels proposed during the experiment.

Table 5. Design and noise variables with their assigned levels.

Variables	**Level 1**	**Level 2**	**Level 3**
A	80	100	120
B	30	60	90
C	0.1	0.2	0.3
D	0.01	0.1	0.2
X	Set1	Set2	Set3
Y	7:3	8:2	9:1
Z	Tr1/Tst1	Tr2/Tst2	Tr3/Tst3

The values for the three levels established in each of the tests regarding the number of neurons for the first hidden layer were $A = 80, 100, \text{and } 120,$ respectively; for the number of neurons in the second hidden layer, they were $B = 30, 60, \text{and } 90,$ respectively; for the constant momentum, they were $C = 0.1, 0.2, \text{and } 0.3,$ respectively; and for the learning rate, they were $D = 0.01, 0.1, \text{and } 0.2,$ respectively.

The values for the initial sets of weights X were randomly determined at all three levels. The values set in the proportions of the dataset for level 1 were $Y = 70\%$ training and 30% testing; for level 2, they were 80% and 20%, and for level 3, they were 90% and 30%, respectively; finally, the random selection of the training and testing set for level

1 was $Z = Training1/Test$; for level 2, it was $Training2/Test2$, and for level 3, it was $Training3/Test3$.

3.2. Experimentation Stage

A total of 36 training sessions were carried out by implementing the OA in $L_9(3^4)$ and $L_4(3^2)$ configurations, where the network architectures were trained and tested, obtaining the results shown in Table 6.

Table 6. Responses measured during the implementation of the crossed OA.

Trial No.	G1	G2	G3	G4	Average	S/N
1	0.0825789	0.08150994	0.08200742	0.08102935	0.0817814	−41.797957
2	0.0651544	0.06734919	0.06683558	0.0666657	0.0665012	−36.960574
3	0.0568985	0.05618239	0.05984281	0.05778079	0.0576761	−31.219083
4	0.074502	0.06984874	0.07645081	0.07592863	0.0741825	−27.856466
5	0.0618532	0.06118982	0.06071765	0.05706847	0.0602073	−28.973660
6	0.0525082	0.05237245	0.05415778	0.05406061	0.0532747	−34.832469
7	0.070358	0.06704642	0.06866684	0.07036554	0.0691092	−32.761969
8	0.0549823	0.04965534	0.05378926	0.05494074	0.0533419	−26.523630
9	0.0465695	0.04604935	0.04942250	0.05026433	0.0480764	−27.302767

For the analysis of the S/N ratio, an analysis of variance (ANOVA) was performed using the statistical software program JMP. The S/N ratio and the mean value of the MSE are two of the specific criteria for determining the appropriate levels in the variables involved in network design, and their choice is determined through a series of validation procedures carried out in the next stage, as described below.

3.3. Analysis Stage

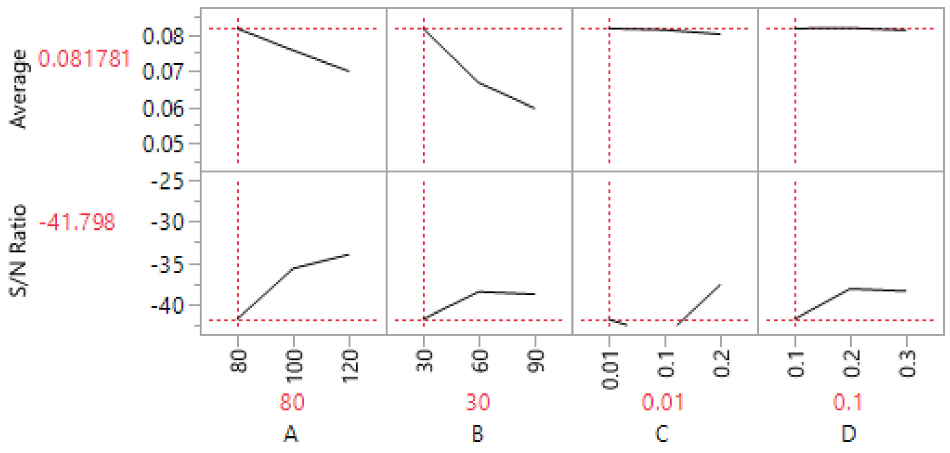
Figure 7a shows the best network topology obtained through the normal profile; Figure 7b describes the best topology through the desirable profile, and Figure 7c describes the best network topology using the maximized desirable profile. The three network profiles were obtained through statistical analysis in the JMP software to identify the optimal values in each of the proposed profiles. After performing the analysis of the S/N ratio, the values in which the levels for each of the variables involved were nearest to the average and S/N ratio red lines on the X axis were chosen, which are described in Table 7.

Table 7. Best design values with normal desirable and maximized profiles.

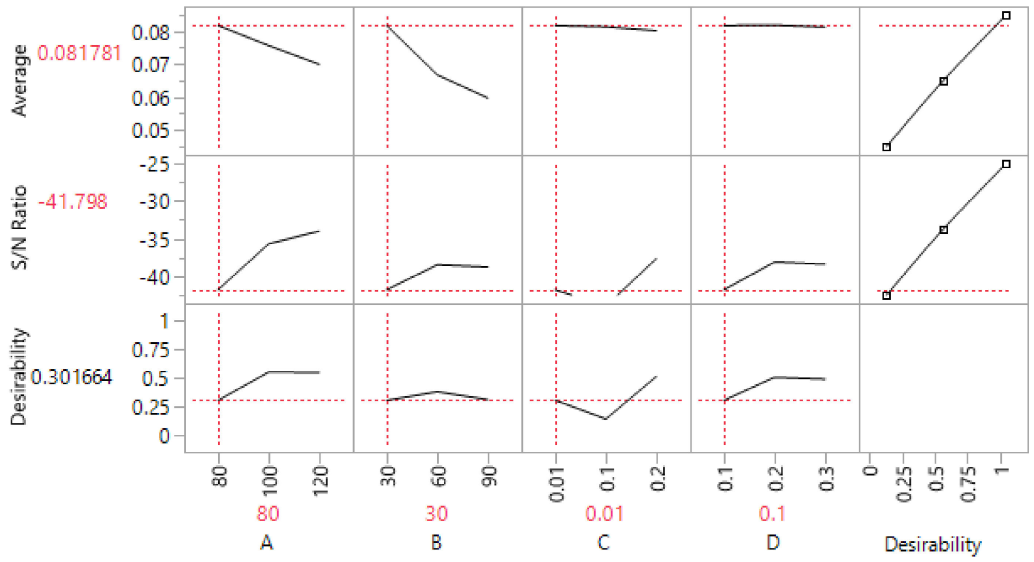
Profile	A	B	C	D
Normal	80	30	0.01	0.1
Desirable	80	30	0.01	0.1
Maximized	100	30	0.2	0.2

For the choice of the best network profile obtained, three training sessions were carried out for each of the three profiles in order to contrast them based on the size of the training and test data and their generalization capacity, estimating the percentage of correct answers in the prediction of the data, obtaining the results shown in Table 8.

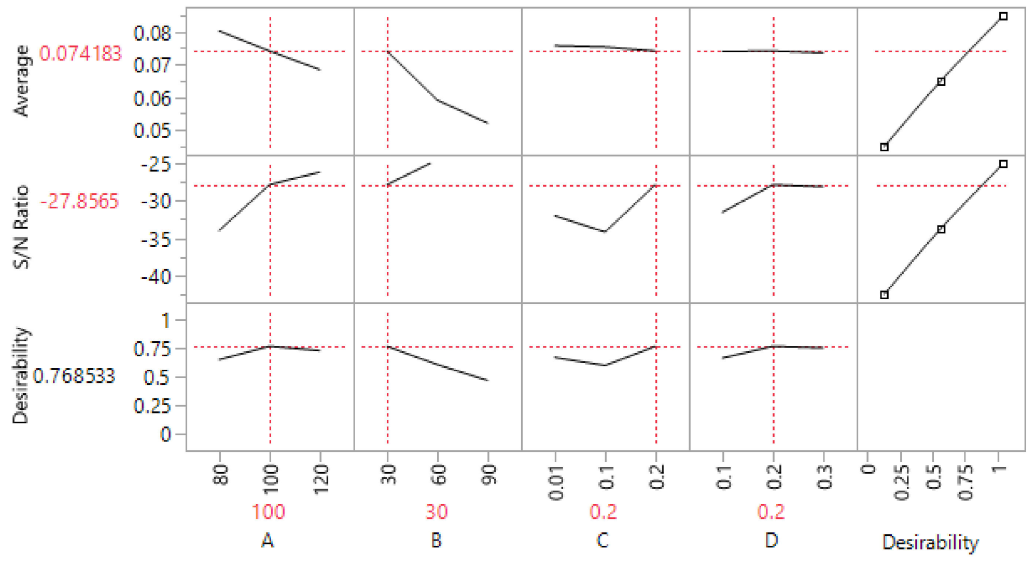
The best topology corresponds to the maximized desirable profile, with the percentage of obtained hits being 87.71% with a margin of error of less than 5% in the tests. Once the best topology was chosen, the statistical tests of correlation and chi-square were performed, showing the best and worst prediction of the network, as shown in Figures 8 and 9, respectively.



(**a**)



(**b**)

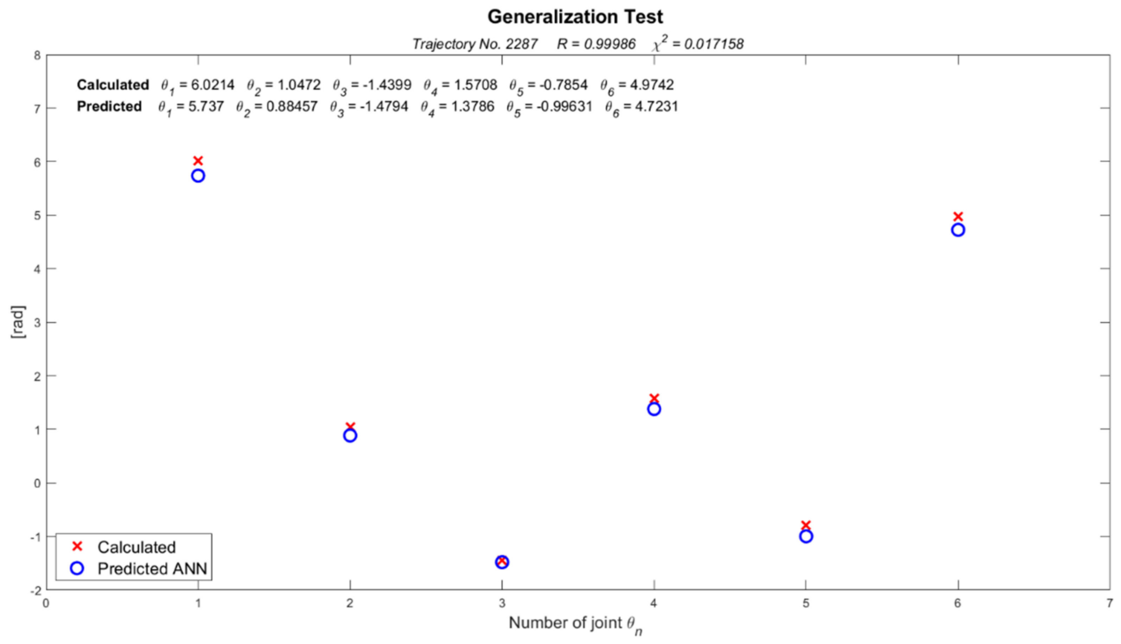


(**c**)

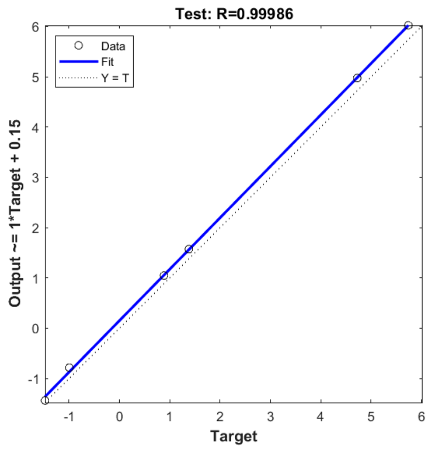
Figure 7. S/N analysis for the determination of the optimal parameters of the network: (**a**) normal profile; (**b**) desirability profile; (**c**) maximized desirability profile.

Table 8. Comparison of density and percentage of hits in the three best profiles.

Training	Profile	Density	Training time (m)	% of Hits χ^2<5%
1	Normal	70:30	13.2649	85.72
2	Normal	80:20	16.1627	85.83
3	Normal	90:10	18.3473	86.40
4	Desirable	70:30	13.4989	87.39
5	Desirable	80:20	16.1765	85.32
6	Desirable	90:10	18.5385	85.50
7	Maximized	70:30	16.3393	87.71
8	Maximized	80:20	17.7275	86.30
9	Maximized	90:10	19.5463	86.86

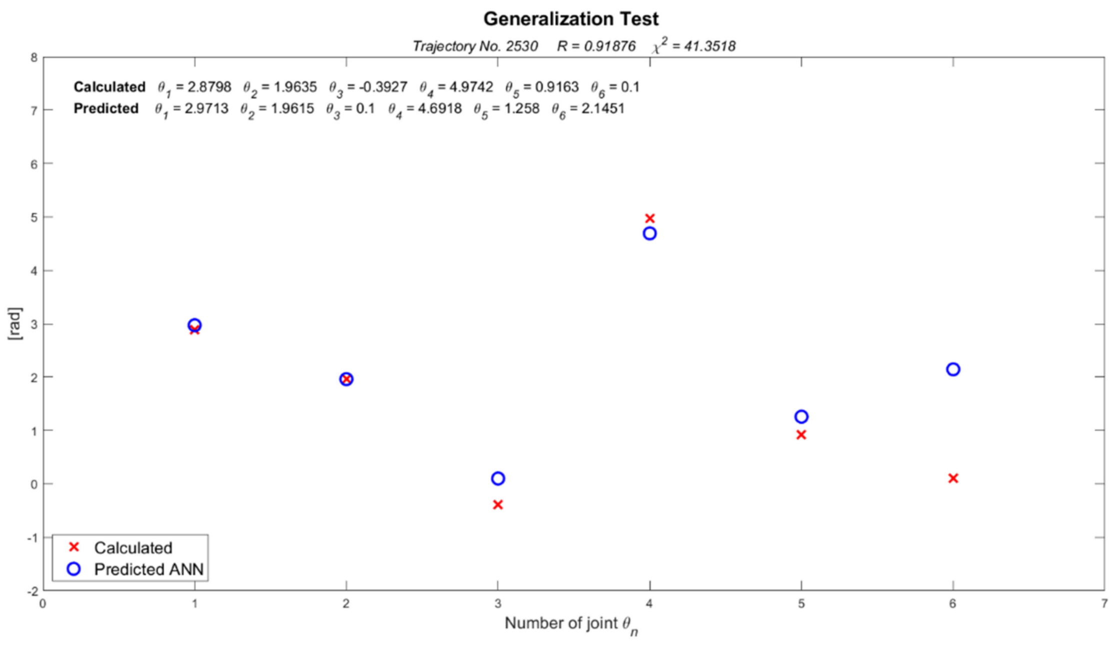


(**a**)

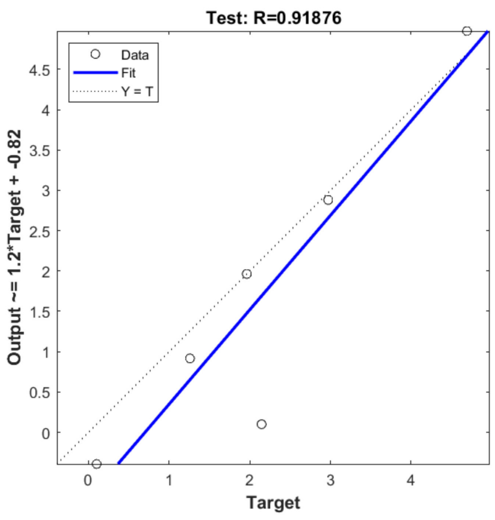


(**b**)

Figure 8. Trajectory of the manipulator with the best prediction and correlation test: (**a**) best predicted values; (**b**) correlation test.



(**a**)



(**b**)

Figure 9. Trajectory of the manipulator with the worst prediction and correlation test: (**a**) worst predicted values; (**b**) correlation test.

To determine if the predicted data are statistically reliable, the cross-validation method was used by splitting the training and testing datasets. The set was split into five subsets of the same size, as shown in Figure 10. The validation subset in each training session was used to measure the generalization error, in other words, the misclassification rate of the model with data dissimilar from those previously applied during the training procedure. The cross-validation procedure was implemented on the training and testing datasets, and the average value of MSE and the standard deviation obtained were very close to those obtained in the confirmation stage [48].

Table 9 shows the results obtained in the cross-validation process, where it is observed that the average training value was equal to 17.5099, the average percentage of hits considering an error of less than 5% was equal to 87.86%, and the average value of MSE was equal to 17.5099, with standard deviations of 1.5848, 1.8974, and 0.0059, respectively.

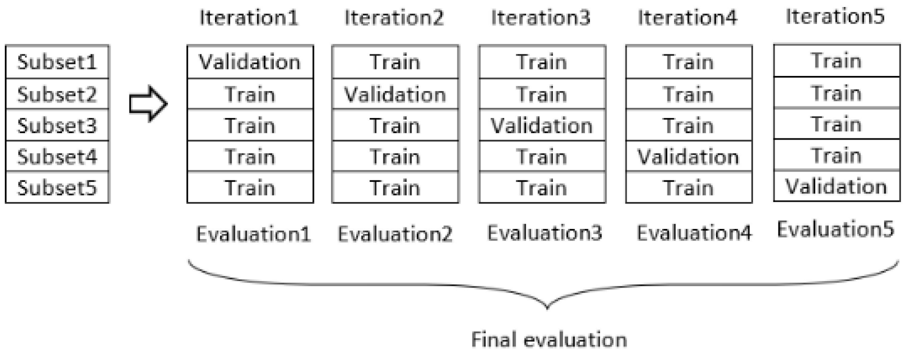


Figure 10. Cross-validation model.

Table 9. Cross-validation results.

Profile	Training Time (m)	% of Hits χ^2<5%	MSE
1	14.6750	85.17	0.0757
2	18.2350	86.95	0.0690
3	18.2165	87.93	0.0669
4	18.2044	89.75	0.0633
5	18.2189	89.51	0.0603
Average	17.5099	87.86	0.0670
Standard deviation	1.5848	1.8974	0.0059

In relation to the three profiles analyzed, the choice of the appropriate levels for the structural parameters of the best network topology were those corresponding to the maximized desirable profile with 100 and 30 neurons, respectively, a momentum of 0.2, and a learning rate of 0.2.

Figure 11 shows the layered surface diagram of the neural network used in this work. The training was performed using MATLAB software. The ANN was composed of an input layer with 100 neurons, a hidden layer with 30 neurons, and an output layer with 6 neurons. All three layers used the activation function. The training algorithm used to adjust the weighting of the synaptic weights was *resilient backpropagation*.

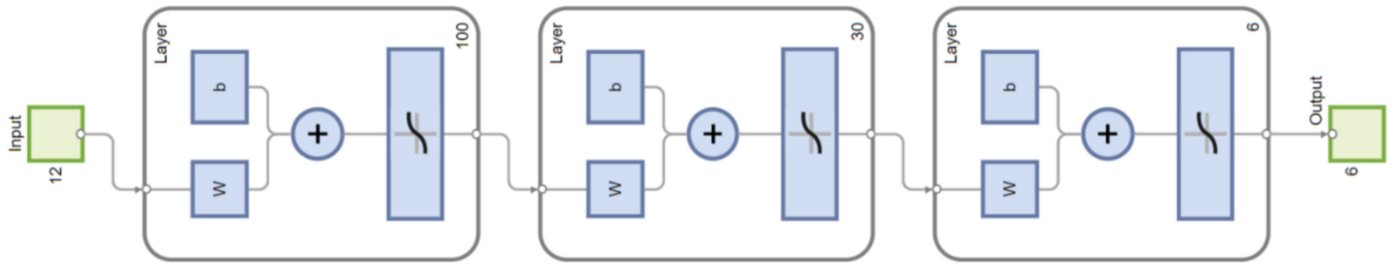


Figure 11. Best maximized desirable topology used in this study.

3.4. Implementation Results Compared with Simulation Results

Table 10 shows the measurement of the 10 trajectories predicted by the Quetzal manipulator and the error generated in comparison with the calculated trajectory. To analyze the data, 10 trajectories were chosen from the training dataset, and the simulation of each of them was carried out in order to obtain the distance traveled from the initial position to the final point.

Table 10. Trajectory comparison.

Trajectory	Calculated Final Position (cm)	Calculated Measure	Predicted Position (cm)	Measured Distance (cm)	Error %
6	[−0.93, −3.81, 59.34]	4.9862	[5.51, 4.22, 58.91]	5.2	4.2
13	[−4.56, −19.98, 52.47]	22.78	[−8.78, −17.90, 52.97]	22.5	1.2
161	[−11.32, −35.76, 22]	55.13	[−27.17, −25.88, 18.96]	54.5	1.1
216	[−22.27, −26.95, 28]	48.97	[−27.18, −23.14, 28]	51.5	5.1
236	[−14.97, −25.94, 25.77]	47.33	[−28.22, −13.01, 20]	51	7.7
317	[14.32, −27.56, 8.46]	62.25	[14.58, −24.46, 4.94]	63.5	2
663	[4.36, −18.87, 46.35]	25.16	[5.12, −19.25, 46.33]	26	3.3
988	[30.75, −12.49, 33.98]	43.70	[26.63, −20.93, 29.13]	45.5	4.1
1025	[−12.41, 1.87, 57.09]	13.63	[−11.11, 4.18, 58.39]	14	2.7
1216	[23.41, 24.49, 37.90]	41.82	[19.59, 28.94, 34.78]	43.5	4
				Average	3.5

The greatest error observed was in trajectory number 236, with a value of 7.7% compared with the calculated one, while for trajectory number 6, the error was 1.1% compared with the calculated one. A mean error of 3.5% was obtained for the implementation of the 10 physically realized trajectories using the low-cost (approximately USD 1500) 3D-printed Quetzal manipulator.

3.5. Comparative Analysis

Table 11 shows the values obtained in the design of the optimized BPNN in comparison with the BPNN based on trial and error and other methods used in the optimization of the structural parameters in ANN. As can be seen, the conventional BPNN method based on trial and error shows a greater difficulty in determining the optimal parameters, whereas the optimized BPNN results in a shorter time in the training process than the other methods; in addition, it involves noise parameters that are necessary to generate greater robustness in the network design.

Table 11. Comparison of results with other methods.

Method	Iterations	Training Time	Total Tested Networks	Prediction Error
BPNN trial and error	Often millions	Several hours	100 in several hours	Undetermined
PSO conventional [49]	5000	Not specified	10	0.00913
PSO [11]	100	03:51:38	10 in 39 h	0.0456
This study	10,000	00:17:30	36 in 178 min	0.0171

4. Conclusions and Discussion

Various approaches and powerful learning algorithms of great value have been introduced in recent decades; however, the integration of the various approaches in ANN optimization has allowed researchers to improve performance and generalizability in ANNs. The results of this work revealed that the proposed systematic and experimental approach is a useful alternative for the robust design of ANNs since it allows simultaneously considering the design and the noise variables, incorporating the concept of robustness in the design of ANNs. The RDANN methodology used in this work was initially proposed in the field of neutron dosimetry, so it was adapted for implementation in the field of robotics, allowing us to improve the performance and generalization capacity in an ANN to find the solution to the inverse kinematics in the Quetzal manipulator.

The time spent during the network design process was significantly reduced compared with the conventional methods based on trial and error. In the methods that are generally proposed by the previous experience of the researcher, the design and modeling of the network can take from several days to a few weeks or even months to test the different ANN architectures, which can lead to a relatively poor design. The use of the RDANN methodology in this study allowed the network design to be carried out in less time, with approximately 13 h of training, due to the orthogonal arrangement corresponding to the 36 training sessions performed using a conventional AMD Ryzen 7 5700 series processor with an integrated graphics card.

Depending on the complexity of the problem, the use of this methodology allows handling times ranging from minutes to hours to determine the best robustness parameters in the network architecture. Therefore, it is possible to streamline the process and reduce efforts, with a high degree of precision in network performance. The use of the RDANN methodology allowed the analysis of the interaction between the values in the design variables that were involved, in order to consider their effects on network performance, thus allowing a reduction in the time and effort spent in the modeling stage and speeding up the selection and interpretation of the optimal values in the structural parameters of the network. The quality of the data in the training sets, without a doubt, can significantly help to increase the performance, generalization capacity, and precision of the results obtained. Although the proposed method was implemented and tested in a low-cost manipulator in this study, in future work, we plan to implement it in an industrial-type robot controller. The implementation of the proposed method in parallel robotic manipulators, where the solution of the kinematics is more complex, is also considered.

Supplementary Materials: The following are available online at https://www.mdpi.com/article/10.3390/app12199512/s1, dataset: dataSetQuetzalRobot.mat, code: workspaceQuetzal.m.

Author Contributions: Conceptualization, T.I.-P. and M.d.R.M.-B.; methodology, J.M.O.-R. and T.I.-P.; software, H.G.-R. and H.A.G.-O.; validation, H.G.-R. and T.I.-P.; formal analysis, J.M.O.-R. and M.d.R.M.-B.; investigation, T.I.-P.; resources, F.O.-D. and H.A.G.-O.; data curation, H.G.-R. and H.A.G.-O.; writing—original draft preparation, T.I.-P.; writing—review and editing, T.I.-P. and F.O.-D.; visualization, H.G.-R. and H.A.G.-O.; supervision, M.d.R.M.-B.; project administration, M.d.R.M.-B.; funding acquisition, J.M.O.-R. and T.I.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Instituto Politécnico Nacional (IPN) under grants number CPE/COTEBAL/14/2021, the Consejo Nacional de Ciencia y Tecnología (CONACYT-BECAS) under grants number 431101/498318 and the Consejo Zacatecano de Ciencia, Tecnología e Innovación (COZCyT).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by the Instituto Politécnico Nacional (IPN) under grants number CPE/COTEBAL/14/2021, the Consejo Nacional de Ciencia y Tecnología (CONACYT-BECAS) under grants number 431101/498318 and the Consejo Zacatecano de Ciencia, Tecnología e Innovación (COZCyT). The authors gratefully acknowledge these support agencies. We sincerely thank the people who provided support and advice for this paper, as well as the reviewers for their comments and suggestions.

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

References

1. Khaw, J.F.C.; Lim, B.S.; Lim, L.E.N. Optimal Design of Neural Networks Using the Taguchi Method. *Neurocomputing* **1995**, *7*, 225–245. [CrossRef]
2. Rankovic, N.; Rankovic, D.; Ivanovic, M.; Lazic, L. A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays. *IEEE Access* **2021**, *9*, 26926–26936. [CrossRef]
3. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*; Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2012; Volume 25.
4. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
5. Kosarac, A.; Mladjenovic, C.; Zeljkovic, M.; Tabakovic, S.; Knezev, M. Neural-Network-Based Approaches for Optimization of Machining Parameters Using Small Dataset. *Materials* **2022**, *15*, 700. [CrossRef] [PubMed]
6. Teslyuk, V.; Kazarian, A.; Kryvinska, N.; Tsmots, I. Optimal Artificial Neural Network Type Selection Method for Usage in Smart House Systems. *Sensors* **2021**, *21*, 47. [CrossRef]
7. Ortiz-Rodríguez, J.M.; Martínez-Blanco, M.d.R.; Viramontes, J.M.C.; Vega-Carrillo, H.R. Robust Design of Artificial Neural Networks Methodology in Neutron Spectrometry. In *Artificial Neural Networks*; Suzuki, K., Ed.; IntechOpen: Rijeka, Croatia, 2013.
8. Zajmi, L.; Ahmed, F.Y.H.; Jaharadak, A.A. Concepts, Methods, and Performances of Particle Swarm Optimization, Backpropagation, and Neural Networks. *Appl. Comput. Intell. Soft Comput.* **2018**, *2018*, 9547212. [CrossRef]
9. Sun, W.; Huang, C. A Carbon Price Prediction Model Based on Secondary Decomposition Algorithm and Optimized Back Propagation Neural Network. *J. Clean. Prod.* **2020**, *243*, 118671. [CrossRef]
10. Huang, D.; Wu, Z. Forecasting Outpatient Visits Using Empirical Mode Decomposition Coupled with Back-Propagation Artificial Neural Networks Optimized by Particle Swarm Optimization. *PLoS ONE* **2017**, *12*, e0172539. [CrossRef]
11. Gaxiola, F.; Melin, P.; Valdez, F.; Castro, J.R.; Castillo, O. Optimization of Type-2 Fuzzy Weights in Backpropagation Learning for Neural Networks Using GAs and PSO. *Appl. Soft Comput.* **2016**, *38*, 860–871. [CrossRef]
12. Huang, H.-X.; Li, J.-C.; Xiao, C.-L. A Proposed Iteration Optimization Approach Integrating Backpropagation Neural Network with Genetic Algorithm. *Expert Syst. Appl.* **2015**, *42*, 146–155. [CrossRef]
13. Amirsadri, S.; Mousavirad, S.J.; Ebrahimpour-Komleh, H. A Levy Flight-Based Grey Wolf Optimizer Combined with Back-Propagation Algorithm for Neural Network Training. *Neural Comput. Appl.* **2018**, *30*, 3707–3720. [CrossRef]
14. Li, S.; Zhang, Y.; Jin, L. Kinematic Control of Redundant Manipulators Using Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2243–2254. [CrossRef]
15. Gao, R. Inverse Kinematics Solution of Robotics Based on Neural Network Algorithms. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 6199–6209. [CrossRef]
16. Kramar, V.; Kramar, O.; Kabanov, A. An Artificial Neural Network Approach for Solving Inverse Kinematics Problem for an Anthropomorphic Manipulator of Robot SAR-401. *Machines* **2022**, *10*, 241. [CrossRef]
17. Šegota, S.B.; Anđelić, N.; Mrzljak, V.; Lorencin, I.; Kuric, I.; Car, Z. Utilization of Multilayer Perceptron for Determining the Inverse Kinematics of an Industrial Robotic Manipulator. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 1729881420925283. [CrossRef]
18. Bai, Y.; Luo, M.; Pang, F. An Algorithm for Solving Robot Inverse Kinematics Based on FOA Optimized BP Neural Network. *Appl. Sci.* **2021**, *11*, 7129. [CrossRef]
19. Jiang, G.; Luo, M.; Bai, K.; Chen, S. A Precise Positioning Method for a Puncture Robot Based on a PSO-Optimized BP Neural Network Algorithm. *Appl. Sci.* **2017**, *7*, 969. [CrossRef]
20. Malik, A.; Lischuk, Y.; Henderson, T.; Prazenica, R. A Deep Reinforcement-Learning Approach for Inverse Kinematics Solution of a High Degree of Freedom Robotic Manipulator. *Robotics* **2022**, *11*, 44. [CrossRef]
21. Malik, A.; Henderson, T.; Prazenica, R. Multi-Objective Swarm Intelligence Trajectory Generation for a 7 Degree of Freedom Robotic Manipulator. *Robotics* **2021**, *10*, 127. [CrossRef]
22. Hassan, A.A.; El-Habrouk, M.; Deghedie, S. Inverse Kinematics of Redundant Manipulators Formulated as Quadratic Programming Optimization Problem Solved Using Recurrent Neural Networks: A Review. *Robotica* **2020**, *38*, 1495–1512. [CrossRef]
23. Zhang, L.; Xiao, N. A Novel Artificial Bee Colony Algorithm for Inverse Kinematics Calculation of 7-DOF Serial Manipulators. *Soft Comput.* **2019**, *23*, 3269–3277. [CrossRef]
24. Bourbonnais, F.; Bigras, P.; Bonev, I.A. Minimum-Time Trajectory Planning and Control of a Pick-and-Place Five-Bar Parallel Robot. *IEEEASME Trans. Mechatron.* **2015**, *20*, 740–749. [CrossRef]
25. Arboretti, R.; Ceccato, R.; Pegoraro, L.; Salmaso, L. Design of Experiments and Machine Learning for Product Innovation: A Systematic Literature Review. *Qual. Reliab. Eng. Int.* **2022**, *38*, 1131–1156. [CrossRef]
26. Ou, Z.; Zhang, M.; Qin, H. Tripling of Fractional Factorial Designs. *J. Stat. Plan. Inference* **2019**, *199*, 151–159. [CrossRef]
27. Rahman, M.A.; Muniyandi, R.c.; Albashish, D.; Rahman, M.M.; Usman, O.L. Artificial Neural Network with Taguchi Method for Robust Classification Model to Improve Classification Accuracy of Breast Cancer. *PeerJ Comput. Sci.* **2021**, *7*, e344. [CrossRef] [PubMed]
28. Manni, A.; Saviano, G.; Bonelli, M.G. Optimization of the ANNs Predictive Capability Using the Taguchi Approach: A Case Study. *Mathematics* **2021**, *9*, 766. [CrossRef]

29. Zhang, Y.; Guo, D.; Li, Z. Common Nature of Learning Between Back-Propagation and Hopfield-Type Neural Networks for Generalized Matrix Inversion With Simplified Models. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 579–592. [CrossRef] [PubMed]
30. Alebooyeh, M.; Urbanic, R.J. Neural Network Model for Identifying Workspace, Forward and Inverse Kinematics of the 7-DOF YuMi 14000 ABB Collaborative Robot. *IFAC PapersOnLine* **2019**, *52*, 176–181. [CrossRef]
31. Köker, R. A Genetic Algorithm Approach to a Neural-Network-Based Inverse Kinematics Solution of Robotic Manipulators Based on Error Minimization. *Inf. Sci.* **2013**, *222*, 528–543. [CrossRef]
32. Volinski, A.; Zaidel, Y.; Shalumov, A.; DeWolf, T.; Supic, L.; Tsur, E.E. Data-Driven Artificial and Spiking Neural Networks for Inverse Kinematics in Neurorobotics. *Patterns* **2022**, *3*, 100391. [CrossRef]
33. Fang, G.; Tian, Y.; Yang, Z.-X.; Geraedts, J.M.P.; Wang, C.C.L. Efficient Jacobian-Based Inverse Kinematics With Sim-to-Real Transfer of Soft Robots by Learning. *IEEEASME Trans. Mechatron.* **2022**, 1–11, *in press*. [CrossRef]
34. Phuoc, L.M.; Martinet, P.; Lee, S.; Kim, H. Damped Least Square Based Genetic Algorithm with Ggaussian Distribution of Damping Factor for Singularity-Robust Inverse Kinematics. *J. Mech. Sci. Technol.* **2008**, *22*, 1330–1338. [CrossRef]
35. Köker, R.; Çakar, T. A Neuro-Genetic-Simulated Annealing Approach to the Inverse Kinematics Solution of Robots: A Simulation Based Study. *Eng. Comput.* **2016**, *32*, 553–565. [CrossRef]
36. Qie, X.; Kang, C.; Zong, G.; Chen, S. Trajectory Planning and Simulation Study of Redundant Robotic Arm for Upper Limb Rehabilitation Based on Back Propagation Neural Network and Genetic Algorithm. *Sensors* **2022**, *22*, 4071. [CrossRef]
37. Liu, R.; Liu, C. Human Motion Prediction Using Adaptable Recurrent Neural Networks and Inverse Kinematics. *IEEE Control Syst. Lett.* **2021**, *5*, 1651–1656. [CrossRef]
38. Reinhart, R.F.; Steil, J.J. Reaching Movement Generation with a Recurrent Neural Network Based on Learning Inverse Kinematics for the Humanoid Robot ICub. In Proceedings of the 2009 9th IEEE-RAS International Conference on Humanoid Robots, Paris, France, 7–10 December 2009; pp. 323–330.
39. Yiyang, L.; Xi, J.; Hongfei, B.; Zhining, W.; Liangliang, S. A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm. *IEEE Access* **2021**, *9*, 32341–32350. [CrossRef]
40. Ghosh, A.; Singh, O.; Ray, A.K. Inverse Kinematic Solution of a 7 DOF Robotic Manipulator Using Boundary Restricted Particle Swarm Optimization. *IFAC PapersOnLine* **2022**, *55*, 101–105. [CrossRef]
41. Wang, D.; Wu, J.; Wang, L.; Liu, Y. A Postprocessing Strategy of a 3-DOF Parallel Tool Head Based on Velocity Control and Coarse Interpolation. *IEEE Trans. Ind. Electron.* **2017**, *65*, 6333–6342.
42. Larrañaga, A. 3D Printable Robotic Arm. Available online: https://github.com/AngelLM (accessed on 5 July 2022).
43. Denavit, J.; Hartenberg, R.S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *J. Appl. Mech.* **1955**, *22*, 215–221. [CrossRef]
44. Corke, P.I.; Khatib, O. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB.*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 73, ISBN 3-642-20144-X.
45. Mostafa, S.A.; Ahmad, I.A. Remainder Linear Systematic Sampling with Multiple Random Starts. *J. Stat. Theory Pract.* **2016**, *10*, 824–851. [CrossRef]
46. Martínez-Blanco, M.d.R.; Ibarra-Pérez, T.; Olivera-Domingo, F.; Ortiz-Rodríguez, J.M. Optimization of Training Data Set Based on Linear Systematic Sampling to Solve the Inverse Kinematics of 6 DOF Robotic Arm with Artificial Neural Networks. In *Frontiers of Data and Knowledge Management for Convergence of ICT, Healthcare, and Telecommunication Services*; Paul, S., Paiva, S., Fu, B., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 85–112. ISBN 978-3-030-77558-2.
47. Gurney, K. *An Introduction to Neural Networks*, 1st ed.; CRC Press: Boca Raton, FL, USA, 1997; ISBN 1-315-27357-8.
48. Paneiro, G.; Rafael, M. Artificial Neural Network with a Cross-Validation Approach to Blast-Induced Ground Vibration Propagation Modeling. *Undergr. Space* **2021**, *6*, 281–289. [CrossRef]
49. Dereli, S.; Köker, R. IW-PSO Approach to the Inverse Kinematics Problem Solution of a 7-DOF Serial Robot Manipulator. *Sigma J. Eng. Nat. Sci.* **2018**, *36*, 77–85.