symmetry

MDPI

Article

Three-Phase Feeder Load Balancing Based Optimized Neural Network Using Smart Meters

Lina Alhmoud [1,*], Qosai Nawafleh [2] and Waled Merrji [3]

[1] Electrical Power Engineering, Yarmouk University, Irbid 21163, Jordan
[2] National Electric Power Company, Substation Maintenance Department, Amman 11181, Jordan; 2016979013@ses.yu.edu.jo
[3] Irbid District Electricity Company, Planning Department, Loads Management Section, Irbid 21100, Jordan; 2017979020@ses.yu.edu.jo
* Correspondence: lina.hmoud@yu.edu.jo

Abstract: The electricity distribution system is the coupling point between the utility and the end-user. Typically, these systems have unbalanced feeders due to the variety of customers' behaviors. Some significant problems occur; the unbalanced loads increase the operational cost and system investment. In radial distribution systems, swapping loads between the three phases is the most effective method for phase balancing. It is performed manually and subjected to load flow equations, capacity, and voltage constraints. Recently, due to smart grids and automated networks, dynamic phase balancing received more attention, thus swapping the loads between the three phases automatically when unbalance exceeds permissible limits by using a remote-controlled phase switch selector/controller. Automatic feeder reconfiguration and phase balancing eliminates the service interruption, enhances energy restoration, and minimize losses. In this paper, a case study from the Irbid district electricity company (IDECO) is presented. Optimal reconfiguration of phase balancing using three techniques: feed-forward back-propagation neural network (FFBPNN), radial basis function neural network (RBFNN), and a hybrid are proposed to control the switching sequence for each connected load. The comparison shows that the hybrid technique yields the best performance. This work is simulated using MATLAB and C programming language.

Keywords: artificial intelligence; feed-forward back-propagation; load balancing; radial basis function

Citation: Alhmoud, L.; Nawafleh, Q.; Marji, W. Three-Phase Feeder Load Balancing Based Optimized Neural Network Using Smart Meters. *Symmetry* **2021**, *13*, 2195. https://doi.org/10.3390/sym13112195

Academic Editor: José Carlos R. Alcantud

Received: 11 October 2021
Accepted: 12 November 2021
Published: 17 November 2021

1. Introduction

The electricity distribution systems are typically unbalanced because of the continuous change in customer loading profile during the day. Once the three phases are not adequately balanced, the risk of over-loading in the network equipment and the power losses increase. Subsequently, the system stability is affected, the supply quality is decreased, and the electricity cost is increased [1]. On the other hand, load balancing improves the reliability and security of the electrical network. Load balancing also minimizes system losses to relieve transformer loading [2,3]. Imbalanced three-phase feeders can be reconfigured by implementing some load balancing techniques such as phase swapping and feeder reconfiguration. The phase swapping technique changes the distribution of loads by swapping them between the phases to make the three phases as equal as possible without changing the feeder topology, as shown in Figures 1 and 2, respectively. Data obtained from the smart meters which are installed at the beginning of each low voltage feeder and customer side are sent directly to the remote controller (brain) to calculate the losses on the feeder. Feeder losses are computed as a difference between the feeder and the summation of the customers' consumption. After that, the controller gives orders to a one-way switching device to take action—if needed—to swipe between the phases feeding customers. Data sent to the controller can be classified into fixed data and variable data, as shown in Figure 3. Fixed data can be introduced as the data regarding the network topology, such as cable

lengths, sizes, and configuration, and customer data such as customer numbers at each connection node. While the variable data are the data that come from the main smart meter fixed at the beginning of the low voltage feeder containing power and energy, and the data coming from the smart meters fixed at the customer side, containing the power and energy consumption, as well.

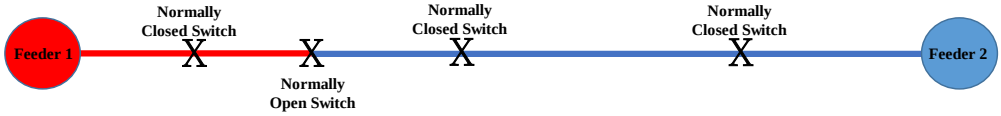

Figure 1. Load balancing before feeder reconfiguration.

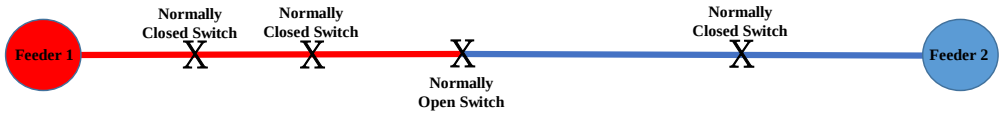

Figure 2. Load balancing after feeder reconfiguration.

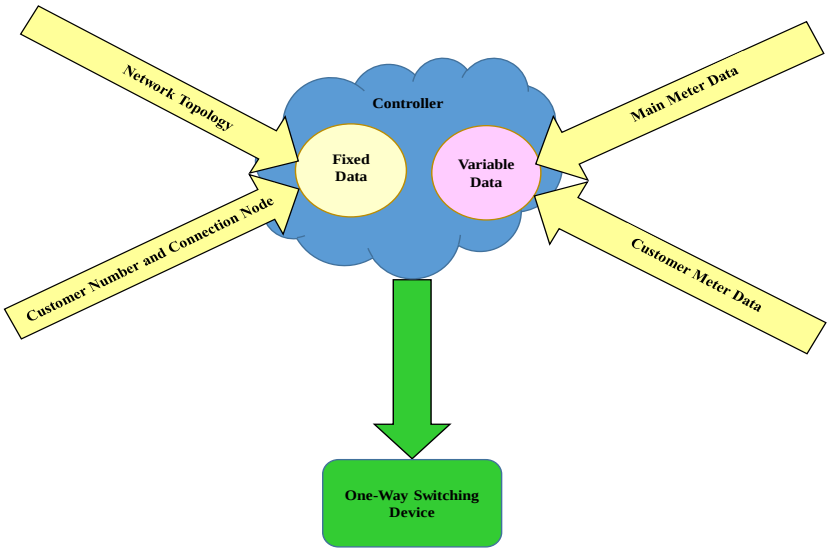

Figure 3. Schematic diagram for data sent to the controller.

This mechanism is implemented using a phase switch selector/controller installed right before the energy meter at the customer side and swapping the customers between the three phases on the same feeder to maintain continual phase balancing. When three phases are connected to the phase switch selector's input side, only one switch should be closed as an output, while the other two phases should remain open, as shown in Figure 4.

The feeder reconfiguration technique changes the feeder topology by moving parts of the feeder to an adjacent one by changing the switches' status so that loads are switched from one feeder to another to relieve the loading of an overloaded feeder to a lightly loaded one [4,5]. Usually, feeder reconfiguration shifts the loads from the heavily loaded feeder to another lightly loaded feeder. Practically, electricity distribution utilities perform load balancing with manual trial and error technique, which is time-consuming, costly, and does not guarantee phase loading equality. Phase balancing automation becomes more realistic, resilient, and agile. It is implemented through power electronics, modern communication techniques, and artificial intelligence. Different types of neural network-based approaches are presented in this work to control the switch selector output to swap customers between the phases. Along with the low voltage distribution feeders, several customarily opened and normally closed switches are distributed to allow transferring load currents between the feeders [6].

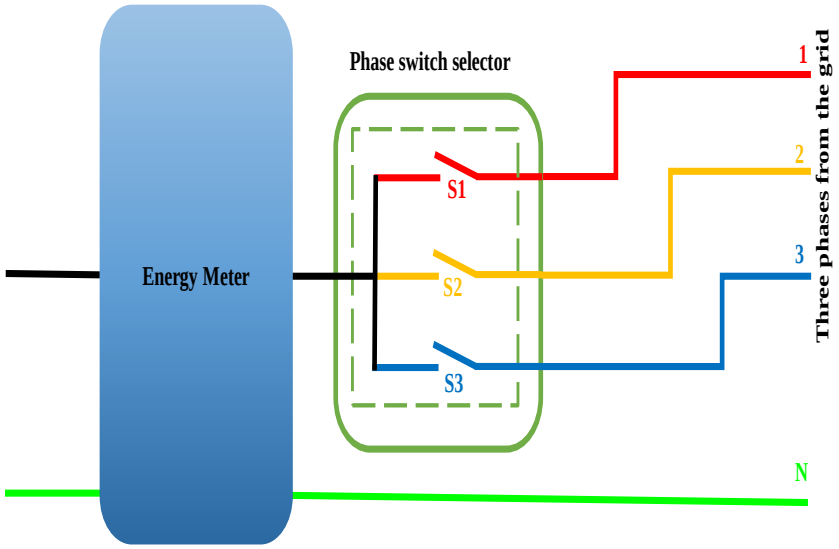

Figure 4. Phase switch selector.

Distribution systems operate under constraints to assure the continuity of supply to the customers under certain quality. The distribution feeders consist of a variety of loads under different categories. This variation in load types and their peak demands do not coincide. Therefore, a variation in loading on some parts of the feeder during the day is noticed. Hence, it is essential to reconfigure the network by rescheduling the loads to operate the system effectively [7]. Feeder reconfiguration modifies the topology of the distribution system by changing the open and close status of switches to better the distribution networks, whereas phase swapping changes the customer connection from one phase to another. The load balancing analysis determines which loads can be reconnected to different phases. Load balancing in the distribution system is defined as preserving the load currents roughly identical to the three phases. Loads are considered evenly distributed on the three phases; i.e., each phase should be connected to 1/3 of the total loads. So, the problem is to find the most appropriate three sets of loads, with minimum differences among the individual sums of the three sets.

The loss minimization in distribution system reconfiguration and load balancing problems of the open-loop radial power are presented using different techniques such as heuristic or meta-heuristic approaches [8–10], mathematical programming [11], and intelligent algorithms [12,13]. The heuristics techniques produce acceptable results with less computation cost. The network reconfiguration with optimal distribution generators siting, sizing, and tie-switch placement for reliability improvement and loss minimization is proposed [14]. The loss minimization using reconfiguration and switching modifications like closing or opening the sectionalizing switches of the distribution feeders are manipulated. A three-phase load balancing using load flow variation technique before electrical installation is presented [15,16]. Load balancing estimation using balancing index calculation is formulated as a non-linear optimization problem with an objective function [17]. Reducing the feeder unbalance using a fuzzy logic is demonstrated [18,19]. Different techniques are carried out to maintain the load balance, such as Ant Colony Optimization [20], support vector machines [21,22], and discrete passive compensator [23]. This work is a real case study for an optimal automatic feeder reconfiguration using three-phase load balancing based artificial neural network (ANN) techniques: radial basis function neural network (RBFNN) [24], feed-forward back-propagation neural network (FFBPNN) [25], and a hybrid. Implementing a hybrid technique is the original contribution. This technique enhances the learning process of FFBPNN, rides over the local minima, speeds the slow rate error convergence, and reforms classification precision.

The rest of this article is organized as follows. Section 2 discusses load balancing. Next, the system techniques under study is presented. ANN techniques are discussed, including RBFNN, FFBPNN, and hybrid techniques are addressed in Section 4. Results and discussion are discussed in Section 5, followed by a conclusion in Section 6.

2. Load Balancing

Distribution network operators face continuous pressure to improve the quality of supply for customers and decrease operating losses. Unbalanced loading of distribution feeders is one of the essential factors affecting low voltage networks' overall losses. The asymmetry factor is high when the overall loading is low, and the asymmetry is significantly smaller during peak load. It means the system is extensively trying to symmetrize the load during the periods with low loading and minimal effect on the overall losses. Thus, the unbalanced loading of the three-phase feeder's distribution and the impact of unbalance currents on the overall losses are considered a hot topic. Electrical utilities modernize power generation and distribution systems. The electric grid transformation offers improved performance and growth opportunities for customers, communities, and businesses. The system deployed in this study is considered the first step towards advanced metering infrastructure that integrates smart meters, software, data centers, and communication networks. Electric companies can enhance their customer services and operations.

The problem is about determining the switches that be opened or closed to obtain load balancing among feeders. Many constraints should be kept, such as voltage drop, thermal constraint, reliability constraint, and capacity constraint of distribution lines and transformers to achieve equal phase loading. The load is dynamic during the day due to the customer's behavior and usage of their appliances. Hence some phases are lightly loaded during a certain period of the day and heavily loaded at another time. Figures 5 and 6 show an example for the unbalanced three-phase loads currents and voltages, respectively. This load is a pure residential load located in Ajlun district. Customers are swapped between the phases continuously to achieve load balancing on the feeder and the transformer. When the smart meters' data are sent to a remote server, it starts to check through optimization techniques if there is any better arrangement for the customers on the three phases to obtain load balancing. If yes, orders are sent to the phase switch selector to swap the customer between phases, with a super-fast action to avoid supply discontinuity. Otherwise, the current situation is the best customer arrangement, and it does nothing [26].

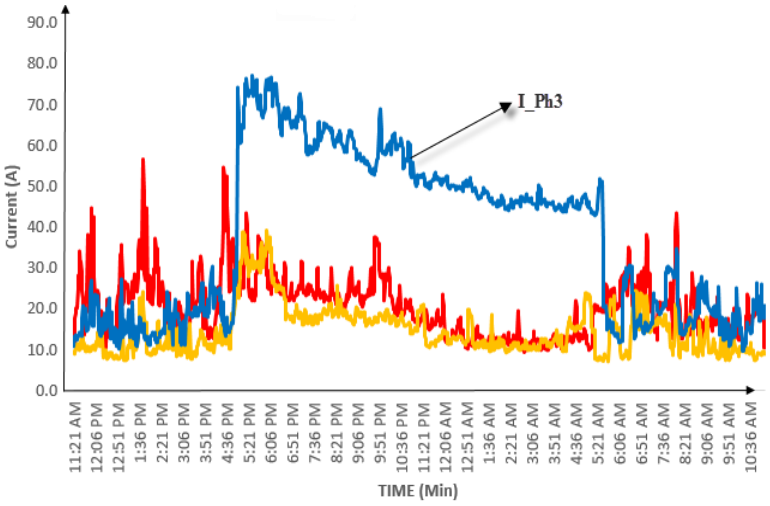

Figure 5. Unbalanced three-phase load currents.

The phase switch selector takes an order from a remote server that collects data from the downstream smart energy meters, calculates the losses at the current situation, and rearranges the loads using one of the proposed algorithms to guarantee the best phase balancing and minimum losses. The new configuration is sent to the phase switch selector

to be implemented. Thus, the phase switch selector takes a three-phase input from the grid; each phase is connected to a switch. One switch of those three switches is closed, while the other should remain open. When an order comes from the server to swap the connected customer from phase A to phase B, a super-fast switching is made to open the switch connected to phase A and close the switch connected to phase B. Discontinuity of supply would not affect the customers because it should be super-fast switching according to the phase switch selector characteristics. This fast-switching time should be mentioned in the datasheet of the phase switch selector and should be fast enough that the customers' appliances would not affect it [27].

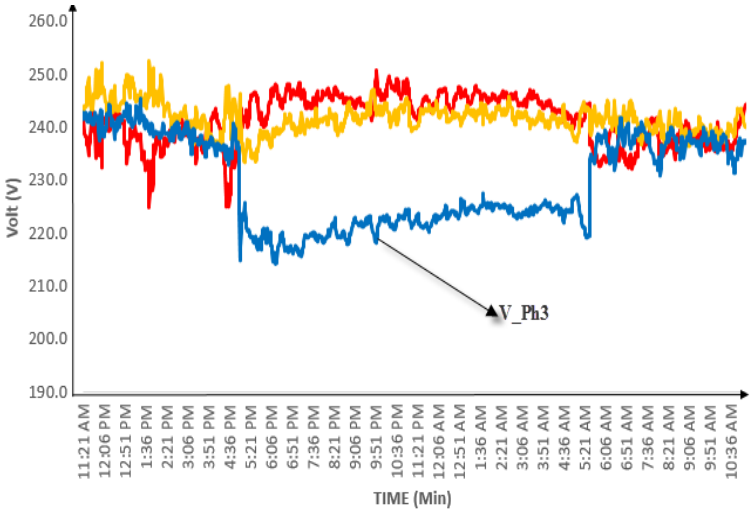

Figure 6. Unbalanced three-phase load voltages.

3. System under Study

In Jordan, distribution feeders are a three-phase, four-wire system. Usually, they are radial or open-loop structures with the same conductor size along the feeder. Balancing loads on a three-phase feeder and reducing neutral current, improving voltage profiles, reducing losses and enhancing system stability and reliability is a very sophisticated task for the utility and engineers because they do not have authority or monitoring over their customers. Practically, phase balancing is carried out manually by trial and error technique based on experience and engineer's knowledge about customer's behavior in that area. By using this manual trial and error method, supply interruption is inevitable when exchanging customers' connection phases to another.

A real case study from IDECO (latitude: 32°33′20.02″ N, longitude: 35°51′0.00″ E) is considered in this work. One of four radial feeders going out from a 630 KVA transformer in the Irbid district is chosen. The feeder under this study has 27 customers, is 470 m in length, and has a 120 mm^2 cross-sectional area. Smart meters are installed along the feeder at the customer side. Their consumption varies from 0.2 kW to 5 kW. Figure 7 shows a schematic diagram for the transformer and the corresponding low voltage network of the four feeders coming out of it, including the feeder under study, while Figure 8 shows a schematic diagram for the same feeder understudy and the number of customers connected to each node. The number of customers equally on the three-phase is not necessary for load balancing, but the current equality on the three phases. In some countries, almost all residential customers have a three-phase connection, but this method is used for single-phase.

Figure 7. Selected transformer for the feeder under study with the corresponding low voltage network.

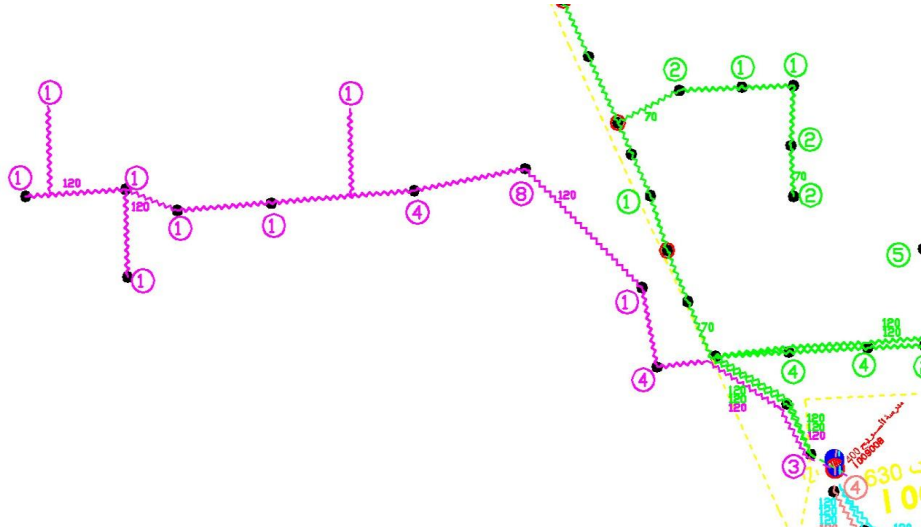

Figure 8. Schematic diagram for the feeder under study.

4. ANN Techniques

RBFNN

RBFNN is an ANN technique that was formulated in 1988 by Broomhead and Lowe. It depends on the linear activation function stored in both input and output layers, whereas the Gaussian activation radial basis function is stored in the hidden layer as shown in Figure 9. However, there are three main parameters; a center that can be determined using clustering techniques, the transfer function, and the distance measured between the input layer and the center. The number of neurons in both input and output layers is determined based on the training pattern, whereas the number of neurons in the hidden layer is determined based on the system's non-linearity. The mathematical model of RBFNN can be represented as follows [28].

$$f(I) = \Psi\left(\frac{\| I - c_i \|}{r_i^2}\right)^2 \quad (1)$$

where I is the input vectors, $f(I)$, r_i, and c_i are the output, radius, and center of ith neurons in the hidden layer, respectively. Ψ is the radial basis function, q is the number of input in the training process and $\| I - c_i \|$ is the distance between the input vectors and the center c_i in the Euclidean space. Clustering technique is used to calculate the center location and is given by Equation (2) [29].

$$\| I - c_i \| = \sqrt{(I_1 - c_{i1})^2 + (I_2 - c_{i2})^2 + \cdots + (I_q - c_{in_i})^2} \quad (2)$$

The width G_{tr} of the basis function (σ) and the weight of the output layer W is given by Equations (3) and (4), respectively [29].

$$G_{tr}\left(\| I_q - c_i \|^2 \right) = e^{\left(\frac{-N_h}{d_{max}} \| I_q - c_i \|^2 \right)} \quad (3)$$

$$W = G_{tr}^{+} \times Y_{tr} \quad (4)$$

The output Y of RBFNN can be obtained using Equation (5) [29].

$$Y = W^T \times G_{tst}^T \quad (5)$$

where G_{tst} is given in Equation (6) and the suffix tst is for testing input vector obtained for certain desired output. Figure 10 shows the main procedure to train RBFNN. It is started with collecting the required data, then selecting the appropriate input and optimum values of the number of neurons in the three layers with their appropriate weight and determining the suitable activation function. Finally, training the network and calculating the error [29].

$$G_{tst}\left(\| I_{tst} - c_i \|^2 \right) = e^{\left(-\frac{\| I_{tst} - c_i \|^2}{2\sigma^2} \right)} \quad (6)$$

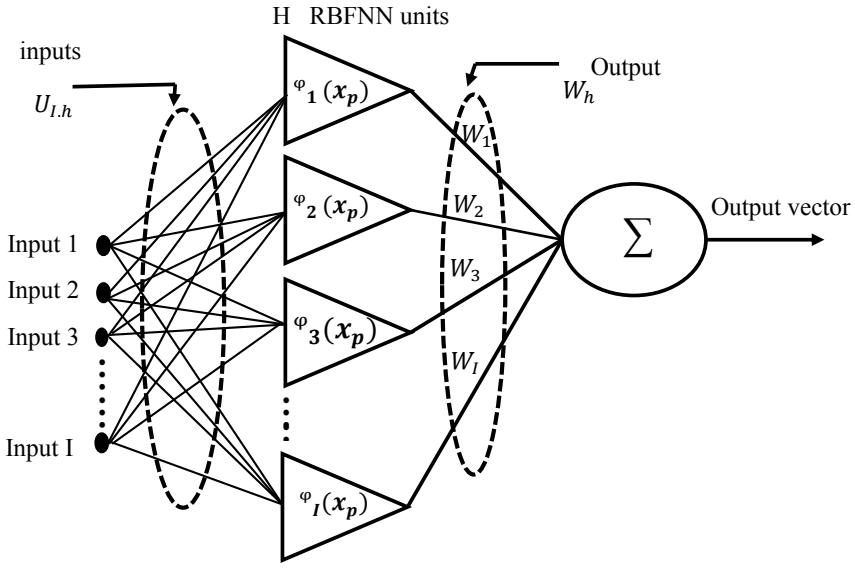

Figure 9. RBFNN architecture [30].

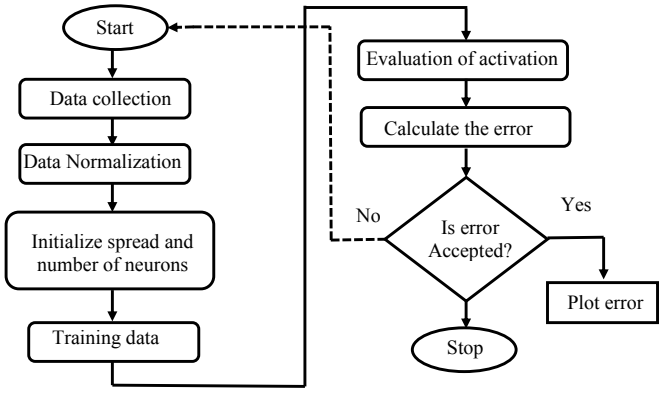

Figure 10. RBFNN flow chart [31].

FFBPNN

Here, the main goal is to minimize the whole network's error by reducing each output neuron's error. The ANN should detect how to map arbitrary input to the output suitably by optimizing the weights. This technique has many features, including accuracy, decreasing the training time, enhancing the processing speed, optimizing the cost function, and improving the mean absolute percentage error (MAP) [31]. The back-propagation algorithm can be described in the FFBPNN technique as shown in Figure 11; the network is created, then the network is trained by giving the input to innovate the output. Thus the network is learned to examine all the values throughout the network. Here, the forward propagation technique can be applied, which means that the input innovates the output, then the backward propagation, including the error being estimated backward towards the input. Lastly, the weight is adjusted, and the error can be reduced by adjusting the weight function. The name of the back-propagation comes from the process sequence[32]. It starts from the input towards the output, then propagates back from the output to the input as shown in Figure 12. Moreover, adjusting the activation functions and the bias through going across. For the first time, these outputs make no sense, but better results are obtained by decreasing error after repeating the process more and more. This algorithm as shown in Figure 13 starts with having new observation $x = [x_1 \ldots x_d]$ and target y^*, then feed forward for each unit g_i in each layer $1 \ldots L$ and compute g_i based on units f_k from previous layer as shown in Equation (7)

$$g_i = \sigma\left(u_{jo} + \sum u_{jk} f_k\right) \tag{7}$$

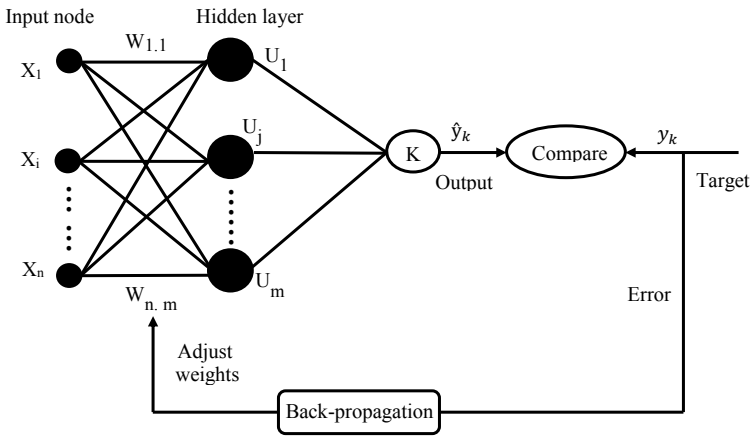

Figure 11. FFBPNN training algorithm with a three-layered architecture [33].

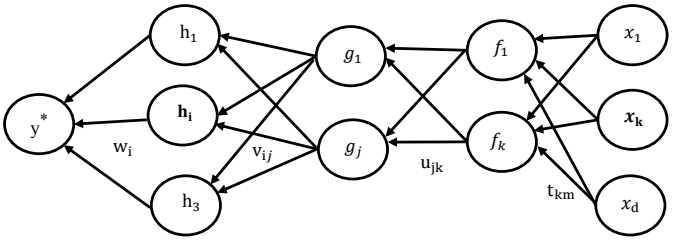

Figure 12. Back-propagation architecture [33].

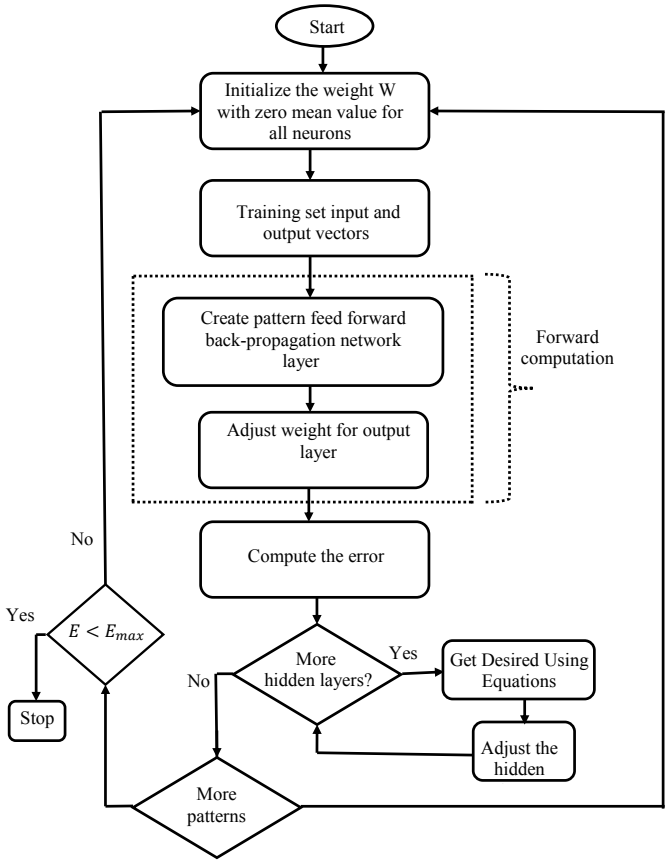

Figure 13. FFBPNN algorithm flow chart [34].

After that, get the prediction y and the target y^2 and calculate the error $(y - y^*)$. For each unit g_i in each layer $L \ldots 1$, an error can be calculated on both g_i and on u_{jk} that affects g_i using Equations (8) and (9), here a sort of synthetic training model is created for all the hidden units in the network, and the errors are propagated by computing the derivative of the error with respect a unit g in the network. The interpretation of the derivative determines to higher or lower of the unit g. Subsequently the g units affects the h units and updating the weight v_{ij}. The nodes are sigmoids and the scaling function $\sigma'(h_i)$ states that H was around zero or one, then whatever changes made to G are not affected. H determines whether G is high or low. Further, the weights f_k that connect G to the nods are higher or lower, as shown in Equation (9). The derivative indicates that the strength is higher or lower. In this logistic process, each iteration, this strength is updated as well as the weight as shown in Equation (10) [33].

$$\frac{\partial E}{\partial g_i} = \sum_i \sigma'(h_i) v_{ij} \frac{\partial E}{\partial h_i} \tag{8}$$

$$\frac{\partial E}{\partial u_{jk}} = \frac{\partial E}{\partial g_i} \sigma' g_j f_k \tag{9}$$

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}} \tag{10}$$

The difference between the output y and the target y^* is calculated using Equation (11). The derivative of the error concerning the unit h_i in the last hidden layers is as shown in Equation (12). The value of derivatives of the error concerning the hidden layers is computed as sigmoid $y(1-y)$ of the bias is added to the weighted product of both the combination of the previously hidden layer and strong connection of the two layers as shown in Equation (13). The derivative of error concerning g is as the same as before. It is shown in Equations (14)–(16), respectively. This nest can be handled as many layers deep as suggested. Clearly, Equations (9)–(16) are the proof for Equation (8) as well as optimizing the error with respect to all of the parameters [34].

$$E = \frac{1}{2}(y - y^*)^2 \tag{11}$$

$$\frac{\partial E}{\partial h_i} = (y - y^*)y(1-y)w_i \tag{12}$$

$$\frac{\partial E}{\partial g_i} = (y - y^*)\frac{\partial y}{\partial g_i} \tag{13}$$

$$\frac{\partial E}{\partial g_i} = (y - y^*)y(1-y)\sum_i w_i \frac{\partial h_i}{\partial g_i} \tag{14}$$

$$\frac{\partial E}{\partial g_i} = (y - y^*)y(1-y)\sum_i w_i h_i(1-h_i)v_{ij} \tag{15}$$

$$\frac{\partial E}{\partial g_i} = \sum_i h_i(1-h_i)v_{ij}\frac{\partial E}{\partial h_i} \tag{16}$$

5. Results and Discussion

The performance of the proposed model is evaluated, different evaluation measures have been adopted, including mean absolute percentage error (*MAPE*), mean squared errors (*MSE*), and the root means squared error (*RMSE*).

- *MAPE*: It shows the deviation of the predicted errors that show how much the predicted points are close to the target line, represented by Equation (17).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

- *MSE*: It is the average of the magnitude of the predicted errors, presented by Equation (18).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{18}$$

- *RMSE*: It shows the deviation of the predicted errors that show how much the predicted points are close to the target line, represented by Equation (19).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{19}$$

where n is the number of observations, $i = 1, 2, 3, \ldots, n$. y_i is the measured value, and $\hat{y}_i$ is the forecasted value. The simulation is executed on an Intel Core i7-8750H CPU, 2.20 GHz, 64.0 GB RAM computer. The proposed ANN is implemented using Mathworks/MATLAB. Different ANN techniques are used, selecting the appropriate number of hidden layers and the number of neurons is the most critical step. This step leads to quick training speed, reduced memory space, and acceptable global generalization capability. The main drawback

of an inappropriate number of hidden nodes may be over-fitting for the input data. The ANN technique is used to solve the load balancing problem. There are around 10,000 samples used as real data obtained from IDECO. Each sample holds current measurements for 27 different loads (houses). It is used to control the switching sequence of each load to keep the three phases balanced. The recorded data are distributed as follows: training set 75%, validation set 10%, and testing set 15%. The ANN inputs are the unbalanced 27 load currents, and the outputs are the switch sequences for each load. The network's output is in the range of (1, 2, 3) for each load. It means the phase number on which switch should be closed or opened for that specific load. The balanced output loads are obtained from implementing a heuristic technique, and they are used to train, test, and validate the ANN. A Matlab/Mathworks command (newff) is used to implement FFBPNN for the whole feeder with an input and output matrices are $(10{,}000 \times 27)$ and $(3 \times 10{,}000)$, respectively. Figure 14 shows the best validation performance for FFBPNN.

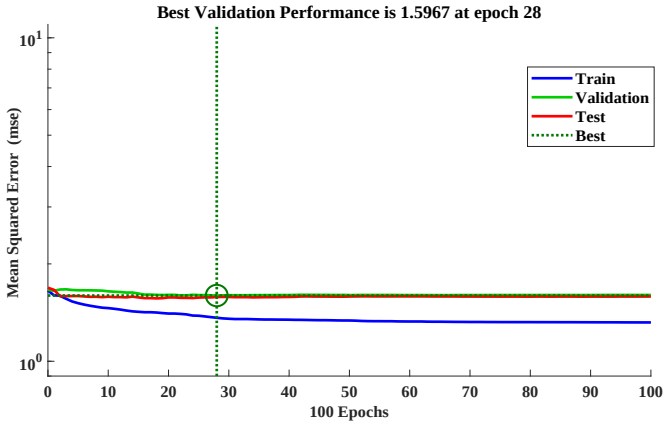

Figure 14. FFBPNN training performance, epoch = 100, and maximum epoch reached.

Tables 1–3 evaluate *MAPE*, *MSE*, and *RMSE* errors, respectively for 10,000 current samples for I_{ph1}, I_{ph2}, and I_{ph3} using FFBPNN for different layer architecture and different iterations (10,100,1000). For example, the first row (10-10-10) means that there are three hidden layers. Each hidden layer contains ten neurons. The optimal performance belongs to the configuration 2000, which means one hidden layer with 2000 neurons. Error evaluations for this technique failed in the load balance test, and therefore it is not recommended in such cases. The distribution of the currents on the three phases was far from the actual currents, and the rate errors were not acceptable. Tables 4–6 evaluate the average error current in terms of spread constant and the number of neurons for 10,000 samples on I_{ph1}, I_{ph2}, and I_{ph3}, respectively using RBFNN. The configuration (5:1000) has optimum evaluation in terms of *MAPE*, *MSE*, and *RMSE*. The errors were 0.15%, 3274, and 57.22% for phase 1, 0.24%, 3560, and 59.67% for phase 2, and 0.15%, 3274, and 57.22% for phase 3, respectively. Tables 7–9 evaluate errors for the three phases using the hybrid technique. This technique has much better results than the two individual techniques in terms of performance and errors calculations.

The configuration (5:1000) has optimum evaluation in terms of *MAPE*, *MSE*, and *RMSE* errors. The results were 0.06%, 664, and 25.75% for phase 1, 0.06%, 663.67, and 25.67% for phase 2, and 0.06%, 678.33, and 26.04% for phase 3, respectively. It is highly recommended for three phases of electrical load balance. Figures 15–17 show the three phases currents via I_{ideal} for the three techniques. Hence, the hybrid technique has the best performance in phase balancing studies. It is practical, flexible, and recommended to IDECO.

Table 1. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph1} using FFBPNN for different layers architecture and iterations.

Hidden Layer Architecture	Average MAPE (%) for I_{ph1} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph1} on Test Set after Each Iteration			Average MSE of Three Iterations	RMSE (%) for I_{ph1} on Test Set after Each Iteration			Average RMSE of Three Iterations
	10	100	1000		10	100	1000		10	100	1000	
10	0.61	0.6	0.58	0.6	7661	7650	7589	7633.33	87.53	87.46	87.11	87.37
10-10	0.63	0.59	0.55	0.59	7700	7601	7500	7600.33	87.75	87.18	86.60	87.18
10-10-10	0.66	0.65	0.53	0.61	7850	7842	7390	7694.00	88.60	88.56	85.97	87.71
100	0.59	0.55	0.50	0.55	7610	7500	7345	7485.00	87.24	86.60	85.70	86.5
100-150	0.62	0.60	0.55	0.59	7690	7650	7500	7613.33	87.69	87.46	86.60	87.25
150-160-170	0.71	0.65	0.54	0.63	8850	7830	7480	8053.33	94.07	88.49	86.49	89.68
1000	0.55	0.53	0.54	0.54	7500	7390	7480	7456.67	86.60	85.97	86.49	86.35
1000-1250	0.6	0.55	0.50	0.55	7650	7501	7345	7498.67	87.46	86.61	85.70	86.59
1500	0.50	0.49	0.49	0.49	7345	7340	7340	7341.67	85.70	85.67	85.67	85.68
1500-1600	0.59	0.62	0.50	0.57	7610	7630	7345	7528.33	87.24	87.35	85.70	86.76
1750	0.50	0.51	0.52	0.51	7346	7455	7380	7393.67	85.71	86.34	85.91	85.99
1750-1750	0.58	0.59	0.53	0.57	7589	7601	7390	7526.67	87.11	87.18	85.97	86.75
2000	**0.49**	**0.47**	**0.45**	**0.47**	**7340**	**7331**	7311	**7327.33**	**85.67**	**85.62**	**85.50**	**85.60**
2000-2000	0.52	0.55	0.47	0.51	7380	7500	7340	7406.67	85.91	86.60	85.67	86.06
2100	0.55	0.59	0.48	0.54	7501	7610	7338	7483.00	86.61	87.24	85.66	86.50
2100-2100	0.60	0.62	0.50	0.57	7650	7689	7345	7561.33	87.46	87.69	85.70	86.95

Table 2. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph2} using FFBPNN for different layers architecture and iterations.

Hidden Layer Architecture	Average MAPE (%) for I_{ph2} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph2} on Test Set after Each Iteration			Average MSE of Three Iterations	RMSE (%) for I_{ph2} on Test Set after Each Iteration			Average RMSE of Three Iterations
	10	100	1000		10	100	1000		10	100	1000	
10	0.97	0.98	0.98	0.97	9870	9901	9901	9890.67	99.35	99.50	99.50	99.45
10-10	0.97	0.99	1	0.987	9870	9985	10,000	9951.67	99.35	99.92	100.00	99.76
10-10-10	1	1	0.98	0.993	10,000	10,000	9901	9967.00	100.00	100.00	99.50	99.83
100	1	1	1	1	10,000	10,000	10,000	10,000	100	100	100	100
100-150	0.98	0.99	0.98	0.983	9901	9985	9901	9929	99.5	99.92	99.5	99.64
150-160-170	0.97	0.98	0.98	0.97	9870	9901	9901	9890.67	99.35	99.50	99.50	99.45
1000	1	0.97	1	0.99	10,000	9870	10,000	9956.67	100.00	99.35	100.00	99.78
1000-1250	0.99	1	0.98	0.99	9985	10,000	9901	9962	99.92	100	99.50	99.81
1500	0.97	1	1	0.99	9870	10,000	10,000	9956.67	99.35	100	100	99.78
1500-1600	0.99	0.98	0.98	0.983	9985	9901	9901	9929	99.92	99.50	99.50	99.64
1750	0.98	1	0.98	0.987	9901	10,000	9901	9934	99.5	100	99.5	99.67
1750-1750	1	1	0.97	0.99	10,000	10,000	9870	9935.00	100	100	99.35	99.78
2000	**0.99**	**0.99**	**0.97**	**0.983**	**9985**	**9985**	**9870**	**9946.67**	**99.92**	**99.92**	**99.35**	**99.73**
2000-2000	1	0.99	0.98	0.99	10,000	9985	9901	9962	100	99.92	99.5	99.81
2100	0.98	0.98	0.99	0.983	9901	9901	9985	9929	99.5	99.5	99.92	99.64
2100-2100	0.99	0.97	1	0.987	9985	9870	10,000	9951.67	99.92	99.35	100	99.76

Table 3. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph3} using FFBPNN for different layers architecture and iterations.

Hidden Layer Architecture	Average MAPE (%) for I_{ph3} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph3} on Test Set after Each Iteration			Average MSE of Three Iterations	RMSE (%) for I_{ph3} on Test Set after Each Iteration			Average RMSE of Three Iterations
	10	100	1000		10	100	1000		10	100	1000	
10	0.6	0.59	0.59	0.59	7650	7610	7610	7623.23	87.46	87.24	87.24	87.31
10-10	061	0.6	0.58	0.6	7661	7650	7589	7633.33	87.53	87.46	87.11	87.37
10-10-10	0.66	0.65	0.65	0.65	7850	7842	7842	7844.67	88.60	88.56	88.56	88.57
100	0.57	0.55	0.60	0.57	7580	7501	7650	7577	87.06	86.61	87.46	87.05
100-150	0.58	0.61	0.61	0.6	7589	7661	7661	7637	87.11	87.53	87.53	87.39
150-160-170	0.67	0.7	0.6	0.66	7856	8847	7650	8117.67	88.63	94.06	87.46	90.05
1000	0.56	0.56	0.54	0.55	7520	7520	7480	7506.67	86.72	86.72	86.49	86.64
1000-1250	0.62	0.62	0.55	0.6	7690	7690	7501	7627	87.69	87.69	86.61	87.33
1500	0.53	0.57	0.60	0.57	7470	7580	7650	7566.67	86.43	87.06	87.46	86.99
1500-1600	0.58	0.59	0.54	0.57	7589	7610	7480	7559.67	87.11	87.24	86.49	86.95
1750	0.56	0.4	0.58	0.51	7520	7214	7589	7441	86.72	84.94	87.11	86.26
1750-1750	0.58	0.59	0.57	0.58	7589	7610	7580	7584.5	87.11	87.23	87.06	87.14
2000	**0.55**	**0.54**	**0.52**	**0.54**	**7501**	**7480**	**7380**	**7453.67**	**86.61**	**86.49**	**85.91**	**86.33**
2000-2000	0.59	0.56	0.55	0.57	7610	7520	7501	7543.67	87.24	86.72	86.61	86.85
2100	0.6	0.6	0.59	0.6	7650	7650	7610	7636.67	87.46	87.46	87.24	87.39
2100-2100	0.65	0.6	0.6	0.62	7842	7650	7650	7714	88.56	87.46	87.46	87.83

Table 4. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph1} using RBFNN for different spread constants and number of neurons.

Speed Constant	No. of Neurons	Average MAPE (%) for I_{ph1}	Average MSE (%) for I_{ph1} on Set Test	RMSE (%) for I_{ph1} on Test Set after Iteration
1	10	0.2	3542	59.51
2	20	0.22	3589	59.91
10	50	0.25	3685	60.7
100	100	0.3	3752	61.25
1000	150	0.33	3789	61.55
1	100	0.24	3560	59.67
2	500	0.18	3489	59.07
5	**1000**	**0.15**	**3274**	**57.22**
10	1500	0.19	3589	59.91
20	2000	0.25	3685	60.7

Table 5. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph2} using RBFNN for different spread constants and number of neurons.

Speed Constant	No. of Neurons	Average MAPE (%) for I_{ph2}	Average MSE (%) for I_{ph2} on Set Test	RMSE (%) for I_{ph2} on Test Set after Iteration
1	10	0.33	3789	61.55
2	20	0.3	3752	61.25
10	50	0.31	3789	61.55
100	100	0.35	3890	62.37
1000	150	0.32	3801	61.65
1	100	0.28	3739	61.15
2	500	0.26	3699	60.82
5	**1000**	**0.24**	**3560**	**59.67**
10	1500	0.27	3701	60.84
20	2000	0.3	3752	61.25

Table 6. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph3} using RBFNN for different spread constants and number of neurons.

Speed Constant	No. of Neurons	Average MAPE (%) for I_{ph3}	Average MSE (%) for I_{ph3} on Set Test	RMSE (%) for I_{ph3} on Test Set after Iteration
1	10	0.21	3502	59.18
2	20	0.22	3589	59.9
10	50	0.26	3699	60.82
100	100	0.29	3785	61.52
1000	150	0.32	3801	61.65
1	100	0.22	3460	58.82
2	500	0.17	3354	57.91
5	**1000**	**0.15**	**3274**	**57.22**
10	1500	0.18	3489	59.07
20	2000	0.26	3699	60.82

Table 7. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph1} using hybrid technique for different spread constants, number of neurons, and iterations.

Spread Constants	No. of Neurons	Average MAPE (%) for I_{ph1} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph1} on Test Set after Each Iteration			Average MSE of Three Iterations	RMSE (%) for I_{ph1} on Test Set after Each Iteration			Average RMSE of Three Iterations
		10	100	1000		10	100	1000		10	100	1000	
1	10	0.11	0.12	0.09	0.11	856	892	747	831.67	29.26	29.87	27.33	28.82
2	20	0.12	0.12	0.09	0.11	892	892	747	843.67	29.87	29.87	27.33	29.02
10	50	0.13	0.13	0.08	0.11	942	941	723	868.67	30.69	30.68	26.89	29.42
100	100	0.11	0.12	0.08	0.1	856	894	723	824.33	29.26	29.9	26.89	28.68
1000	150	0.09	0.1	0.07	0.08	747	842	699	762.67	27.33	29.02	26.44	27.6
1	100	0.09	0.09	0.06	0.08	747	749	682	726	27.33	27.37	26.12	26.94
2	500	0.08	0.08	0.05	0.07	723	724	643	696.67	26.91	26.91	25.36	26.39
5	**1000**	**0.06**	**0.07**	**0.04**	**0.06**	**701**	**689**	**601**	**663.67**	**26.25**	**26.25**	**24.52**	**25.67**
10	1500	0.09	0.09	0.07	0.08	747	746	699	730.67	27.31	27.31	26.44	27.02
20	2000	0.11	0.12	0.09	0.1	856	892	745	831	29.87	29.87	27.29	29.01

Table 8. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph2} using hybrid technique for different spread constants, number of neurons, and iterations.

Spread Constants	No. of Neurons	Average MAPE (%) for I_{ph2} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph2} on Test Set after Each Iteration			Average MSE of of Three Iterations	RMSE (%) for I_{ph2} on Test Set after Each Iteration			Average RMSE of Three Iterations
		10	100	1000		10	100	1000		10	100	1000	
1	10	0.22	0.22	0.1	0.18	1821	1821	842	1494.67	42.67	42.67	29.02	38.12
2	20	0.2	0.22	0.1	0.17	1224	1821	842	1295.67	42.67	42.67	29.02	38.12
10	50	0.2	0.21	0.08	0.16	1242	1412	723	1125.67	37.58	37.58	26.89	34.01
100	100	0.18	0.17	0.07	0.14	1105	1022	699	942	31.97	31.97	26.44	30.13
1000	150	0.14	0.13	0.07	0.11	901	940	699	846.67	30.66	30.66	26.44	29.25
1	100	0.1	0.12	0.05	0.09	842	892	642	792	29.87	29.87	25.34	28.36
2	500	0.08	0.08	0.05	0.07	723	723	642	696	26.89	26.89	25.34	26.37
5	**1000**	**0.06**	**0.07**	**0.04**	**0.06**	**701**	**689**	**601**	**663.67**	**26.25**	**26.25**	**24.52**	**25.67**
10	1500	0.09	0.09	0.07	0.08	747	746	699	730.67	27.31	27.31	26.44	27.02
20	2000	0.11	0.12	0.09	0.1	856	892	745	831	29.87	29.87	27.29	29.01

Table 9. Error evaluations in terms of *MAPE*, *MSE*, and *RMSE* for I_{ph3} using hybrid technique for different spread constants, number of neurons, and iterations.

Spread Constants	No. of Neurons	Average MAPE (%) for I_{ph3} on Test Set after Each Iteration			Average MAPE of Three Iterations	Average MSE (%) for I_{ph3} on Test Set after Each Iteration			Average MSE of Three Iterations	RMSE (%) for I_{ph3} on Test Set after Each Iteration			Average RMSE of Three Iterations
		10	100	1000		10	100	1000		10	100	1000	
1	10	0.12	0.12	0.13	0.12	892	892	941	908.33	29.87	29.87	30.68	30.14
2	20	0.13	0.12	0.11	0.12	941	892	856	896.33	30.68	29.87	29.26	29.93
10	50	0.11	0.12	0.11	0.11	856	892	856	868	29.26	29.87	29.26	29.46
100	100	0.1	0.1	0.11	0.1	842	842	856	846.67	29.02	29.02	29.26	29.1
1000	150	0.1	0.09	0.09	0.09	842	747	747	778.67	29.02	27.33	27.33	27.89
1	100	0.09	0.08	0.08	0.08	747	723	723	731	27.33	26.89	26.89	27.04
2	500	0.08	0.08	0.06	0.07	723	723	701	715.67	26.89	26.89	26.48	26.75
5	**1000**	**0.07**	**0.06**	**0.05**	**0.06**	**689**	**701**	**645**	**678.33**	**26.25**	**26.48**	**25.4**	**26.04**
10	1500	0.1	0.08	0.08	0.09	842	723	723	762.67	29.02	26.89	26.89	27.6
20	2000	0.12	0.12	0.11	0.12	892	892	856	880	29.87	29.87	27.26	29.66

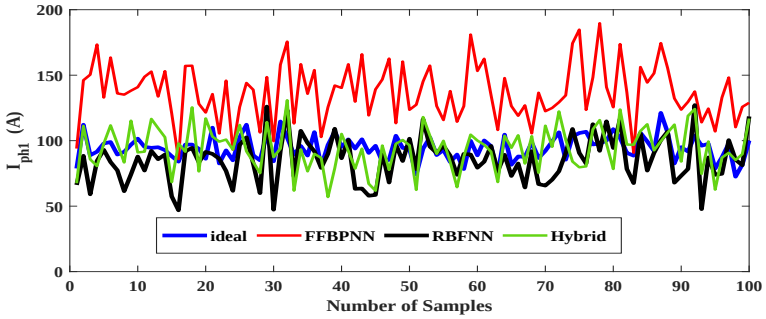

Figure 15. I_{ph1} via I_{ideal} for different samples and techniques.

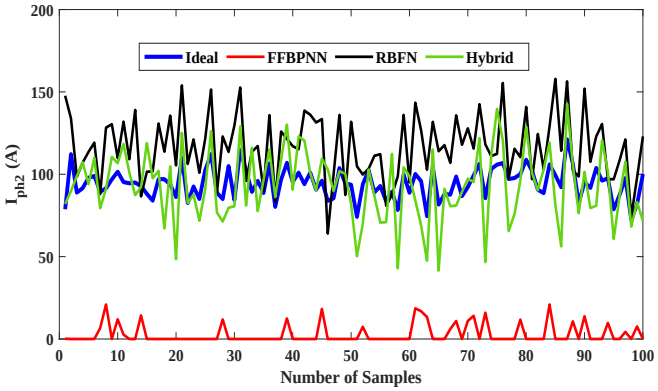

Figure 16. I_{ph2} via I_{ideal} for different samples and techniques.

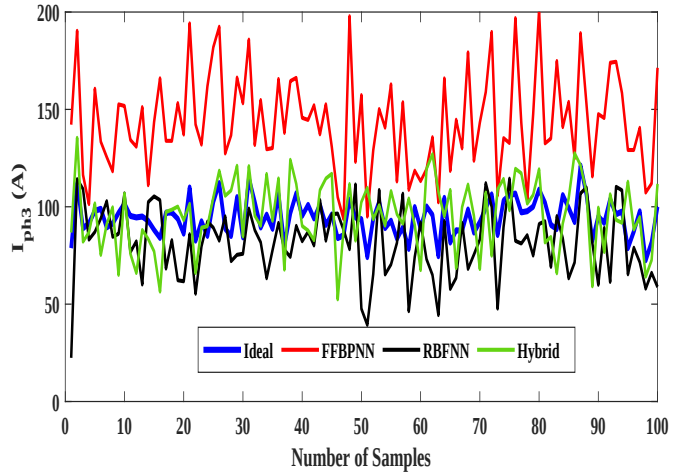

Figure 17. I_{ph3} via I_{ideal} for different samples and techniques.

6. Conclusions

This work presents a MATLAB-based solution for ANN techniques for load balancing investigation. These techniques are successfully tested and validated using simulated real data. The testing results obtained show that the FFBPNN technique has a significant deviation from the desired ideal current. The results obtained using this technique are the worst. On the other hand, the hybrid technique is more guaranteed to give analytical results for load balancing problems than using FFBPNN or RBFNN techniques individually. It showed a better convergence, faster training, and classification thoroughness using a discrete data set; moreover, the results were very close to the ideal values of currents and the acceptable error profiles. This technique is considered the most operative, and it is highly recommended that IDECO use it in load balancing studies since the three-phase balancing has many advantages for customers and utilities.

Author Contributions: Conceptualization, L.A., Q.N. and W.M.; methodology, L.A., Q.N. and W.M.; software, Q.N.; validation, L.A., Q.N. and W.M.; investigation, Q.N.; resources, Q.N. and W.M.; data curation, W.M.; writing—original draft preparation, L.A. and W.M.; writing—review and editing, L.A. and W.M.; visualization, L.A. and Q.N.; supervision, L.A., project administration, L.A., Q.N. and W.M. 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: Not applicable.

Acknowledgments: The authors would like to express special thanks to IDECO for providing access to the research data.

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

Abbreviations

ANN	Artificial Neural Network
FFBPNN	Feed-forward Back-Propagation Neural Network
IDECO	Irbid Distribution Electricity Company
MAPE	Mean Squared Percentage Error
MSE	Mean Square Error
RBFNN	Radial Basis Function Neural Network
RMSE	Root Mean Squared Error

References

1. Kong, W.; Ma, K.; Fang, L.; Wei, R.; Li, F. Cost-Benefit Analysis of Phase Balancing Solution for Data-Scarce LV Networks by Cluster-Wise Gaussian Process Regression. *Power Syst. IEEE Trans.* **2020**, *35*, 3170–3180. [CrossRef]
2. Zheng, W.; Huang, W.; Hill, D.J.; Hou, Y. An adaptive distributionally robust model for three-phase distribution network reconfiguration. *IEEE Trans. Smart Grid* **2020**, *12*, 1224–1237. [CrossRef]
3. Civanlar, S.; Grainger, J.J.; Yin, H.; Lee, S.S.H. Distribution feeder reconfiguration for loss reduction. *IEEE Trans. Power Deliv.* **1988**, *3*, 1217–1223. [CrossRef]
4. Wang, W.; Yu, N. Maximum marginal likelihood estimation of phase connections in power distribution systems. *IEEE Trans. Power Syst.* **2020**, *35*, 3906–3917. [CrossRef]
5. Ukil, A.; Siti, M.; Jordaan, J. Feeder load balancing using combinatorial optimization-based heuristic method. In Proceedings of the 2008 13th International Conference on Harmonics and Quality of Power, Wollongong, NSW, Australia, 28 September–1 October 2008; pp. 1–6. [CrossRef]
6. Chen, C.S.; Cho, M.Y. Energy loss reduction by critical switches. *IEEE Trans. Power Deliv.* **1993**, *8*, 1246–1253. [CrossRef]
7. Ma, K.; Fang, L.; Kong, W. Review of distribution network phase unbalance: Scale, causes, consequences, solutions, and future research directions. *CSEE J. Power Energy Syst.* **2020**, *6*, 479–488.
8. Ganesh, S.; Kanimozhi, R. Meta-heuristic technique for network reconfiguration in distribution system with photovoltaic and D-STATCOM. *IET Gener. Transm. Distrib.* **2018**, *12*, 4524–4535. [CrossRef]
9. Al-Kharsan, I.H.; Marhoon, A.F.; Mahmood, J.R. A New Strategy for Phase Swapping Load Balancing Relying on a Meta-Heuristic MOGWO. *Algorithm. J. Mech. Contin. Math. Sci* **2020**, *15*, 84–102. [CrossRef]
10. Jena, U.K.; Das, P.K.; Kabat, M.R. Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *J. King Saud-Univ.-Comput. Inf. Sci.* **2020**, in press. [CrossRef]
11. Grigoraș, G.; Neagu, B.C.; Gavrilaș, M.; Triștiu, I.; Bulac, C. Optimal phase load balancing in low voltage distribution networks using a smart meter data-based algorithm. *Mathematics* **2020**, *8*, 549. [CrossRef]
12. Kocarev, L.; Zdraveski, V.; Todorovski, M. Method and System for Dynamic Intelligent Load Balancing. U.S. Patent 10,218,179, 26 February 2019.
13. Azizivahed, A.; Narimani, H.; Naderi, E.; Fathi, M.; Narimani, M.R. A hybrid evolutionary algorithm for secure multi-objective distribution feeder reconfiguration. *Energy* **2017**, *138*, 355–373. [CrossRef]
14. Fu, L.; Liu, B.; Meng, K.; Dong, Z.Y. Optimal restoration of an unbalanced distribution system into multiple microgrids considering three-phase demand-side management. *IEEE Trans. Power Syst.* **2020**, *36*, 1350–1361. [CrossRef]
15. Aprilia, E.; Meng, K.; Zeineldin, H.H.; Al Hosani, M.; Dong, Z.Y. Modeling of distributed generators and converters control for power flow analysis of networked islanded hybrid microgrids. *Electr. Power Syst. Res.* **2020**, *184*, 106343. [CrossRef]

16. Ramadhani, U.H.; Shepero, M.; Munkhammar, J.; Widén, J.; Etherden, N. Review of probabilistic load flow approaches for power distribution systems with photovoltaic generation and electric vehicle charging. *Int. J. Electr. Power Energy Syst.* **2020**, *120*, 106003. [CrossRef]
17. Lin, W.M.; Chin, H.C. A current index based load balance technique for distribution systems. In Proceedings of the POWERCON'98. 1998 International Conference on Power System Technology. Proceedings (Cat. No. 98EX151), Beijing, China, 18–21 August 1998; Volume 1, pp. 223–227.
18. Huang, M. A Receiver-Initiated Approach with Fuzzy Logic Control in Load Balancing. *J. Comput. Commun.* **2020**, *8*, 107–119. [CrossRef]
19. Juneja, K. A fuzzy-controlled differential evolution integrated static synchronous series compensator to enhance power system stability. *IETE J. Res.* **2020**. [CrossRef]
20. Torkzadeh, S.; Soltanizadeh, H.; Orouji, A.A. Energy-aware routing considering load balancing for SDN: A minimum graph-based Ant Colony Optimization. *Clust. Comput.* **2021**, *24*, 2293–2312. [CrossRef]
21. Siti, M.W.; Jimoh, A.A.; Jordaan, J.A.; Nicolae, D.V. The Use of Support Vector Machine for Phase Balancing in the Distribution Feeder. In Proceedings of the International Conference on Neural Information Processing 2007, Kitakyushu, Japan, 13–16 November 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 721–729.
22. Jordaan, J.A.; Siti, M.W.; Jimoh, A.A. Distribution feeder load balancing using support vector machines. In Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Daejeon, Korea, 2–5 November 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 65–71.
23. Xuan Tung, N.; Fujita, G.; Horikoshi, K. Phase load balancing in distribution power system using discrete passive compensator. *IEEJ Trans. Electr. Electron. Eng.* **2010**, *5*, 539–547. [CrossRef]
24. Baghaee, H.R.; Mirsalim, M.; Gharehpetian, G.B.; Talebi, H.A. Unbalanced harmonic power sharing and voltage compensation of microgrids using radial basis function neural network-based harmonic power-flow calculations for distributed and decentralised control structures. *IET Gener. Transm. Distrib.* **2017**, *12*, 1518–1530. [CrossRef]
25. Tabatabaei, S. A probabilistic neural network based approach for predicting the output power of wind turbines. *J. Exp. Theor. Artif. Intell.* **2017**, *29*, 273–285. [CrossRef]
26. Ivanov, O.; Neagu, B.; Gavrilas, M.; Grigoras, G.; Sfintes, C. Phase Load Balancing in Low Voltage Distribution Networks Using Metaheuristic Algorithms. In Proceedings of the 2019 International Conference on Electromechanical and Energy Systems (SIELMEN), Craiova, Romania, 9–11 October 2019; pp. 1–6. [CrossRef]
27. Gavrilas, M. Heuristic and metaheuristic optimization techniques with application to power systems. In Proceedings of the 12th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering, Stevens Point, WI, USA, 21–23 October 2010; pp. 95–103.
28. Nicolae, D.V.; Siti, M.W.; Jimoh, A.A. LV self balancing distribution network reconfiguration for minimum losses. In Proceedings of the 2009 IEEE Bucharest PowerTech, Bucharest, Romania, 28 June–2 July 2009; pp. 1–6. [CrossRef]
29. Singh, N.K.; Tripathy, M.; Singh, A.K. A radial basis function neural network approach for multi-hour short term load-price forecasting with type of day parameter. In Proceedings of the 2011 6th International Conference on Industrial and Information Systems, Kandy, Sri Lanka, 16–19 August 2011; pp. 316–321. [CrossRef]
30. Cecati, C.; Kolbusz, J.; Różycki, P.; Siano, P.; Wilamowski, B.M. A Novel RBF Training Algorithm for Short-Term Electric Load Forecasting and Comparative Studies. *IEEE Trans. Ind. Electron.* **2015**, *62*, 6519–6529. [CrossRef]
31. Shafie, A.H.E.; El-Shafie, A.; Almukhtar, A.; Taha, M.R.; Mazoghi, H.G.E.; Shehata, A. Radial basis function neural networks for reliably forecasting rainfall. *J. Water Clim. Chang.* **2012**, *3*, 125–138. [CrossRef]
32. Masoumi, A.; Jabari, F.; Zadeh, S.G.; Mohammadi-Ivatloo, B. Long-Term Load Forecasting Approach Using Dynamic Feed-Forward Back-Propagation Artificial Neural Network. In *Optimization of Power System Problems*; Springer: Cham, Switzerland, 2020; pp. 233–257.
33. Albaradeyia, I.; Hani, A.; Shahrour, I. WEPP and ANN models for simulating soil loss and runoff in a semi-arid Mediterranean region. *Environ. Monit Assess.* **2011**, *180*, 537–556. [CrossRef] [PubMed]
34. Gupta, D.K.; Kumar, P.; Mishra, V.N.; Prasad, R.; Dikshit, P.K.S.; Dwivedi, S.B.; Ohri, A.; Singh, R.S.; Srivastava, V. Bistatic measurements for the estimation of rice crop variables using artificial neural network. *Adv. Space Res.* **2015**, *55*, 1613–1623. [CrossRef]