RBRH

Revista Brasileira de Recursos Hídricos
Brazilian Journal of Water Resources
Versão On-line ISSN 2318-0331
RBRH, Porto Alegre, v. 26, e37, 2021
Scientific/Technical Article

https://doi.org/10.1590/2318-0331.262120210100

Optimal architecture for artificial neural networks as pressure estimator

Arquitetura ótima para redes neurais artificiais como estimadoras de pressão

Rui Gabriel Modesto de Souza[1,2], Bruno Melo Brentan[1] & Gustavo Meirelles Lima[1]

[1]Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil
[2]Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte, MG, Brasil
E-mails: rui.g182@gmail.com (RGMS), brentan@ehr.ufmg.br (BMB), gustavo.meirelles@ehr.ufmg.br (GML)

Received: July 08, 2021 - Revised: October 15, 2021 - Accepted: October 18, 2021

ABSTRACT

The knowledge of hydraulic parameters in water distribution networks can indicate problems in real time, such as pipe bursts, small leakages, increase in pipe roughness and illegal connections. However, an accurate indication relies on the quantity and quality of the data acquired, i.e., the number of sensors used to monitor the network and their location. It is not economic feasible have a great number of sensors, thus, the use of artificial intelligence, such as Artificial Neural Networks (ANNs) can reduce the lack of information necessary to identify problems, estimating hydraulic parameter through the few information collected. The reliability of ANNs depends on its architecture, so, in this paper, different conditions are tested for ANN training to identify which are the most relevant parameters to be adjusted when the ANN is used for pressure estimation.

Keywords: Artificial neural network; Water distribution network; Data-driven model.

RESUMO

O conhecimento dos parâmetros hidráulicos em uma rede de distribuição de água pode ser usado para indicar problemas em tempo real, como rompimentos de tubos, pequenos vazamentos, aumento da rugosidade do tubo e conexões ilegais. No entanto, uma indicação precisa depende da quantidade e qualidade dos dados adquiridos, ou seja, do número de sensores usados para monitorar a rede e sua localização. Entretanto, não é economicamente viável ter um grande número de sensores, portanto, o uso da inteligência artificial, como Redes Neurais Artificiais (RNAs) podem reduzir a falta de informações necessárias para identificar problemas, estimando parâmetros hidráulicos através das poucas informações coletadas. A confiabilidade das RNAs depende de sua arquitetura, então, neste trabalho, diferentes condições são testadas para o treinamento das RNAs de modo a identificar quais são os parâmetros mais relevantes a serem ajustados quando utilizadas na estimativa de pressão.

Palavras-chave: Rede neural artificial; Rede de distribuição de água; Modelos guiados por dados.

INTRODUCTION

A Water Distribution Network (WDN) is composed by a variety of hydraulic components, such as pipes, pumps, valves and tanks. Each of them has a particular purpose in the WDN operation. However, they are all integrated in the same system, and any change in one of these components reflects in all others. Add to this the highly time variant hydraulic conditions, established by the varying water consumption through the day. The result is a complex and nonlinear hydraulic model, but able to provide valuable information to achieve an efficient operation.

In general, WDNs aims to operate with minimum costs, with energy consumption and leakage as two of the main parameters to be minimized. Thus, different approaches can be used, such as: set an optimal rotational speed for pumps according to the hydraulic parameters (Brentan et al., 2018); operate pressure reducing valves to control pressure and reduce leakages (Fontana et al., 2018); use tanks to reduce pump stations operation in peak hours (Xu et al., 2015); and use microturbines to recovery energy excess (Fecarotta et al., 2018).

In recent years, a special attention has been given to the real-time operation. As highlighted by Salomons & Housh (2020), two main blocks are necessary: a demand predictor, to establish the future conditions that has to be satisfied, and an optimization model coupled with a hydraulic simulator, to determine the optimal operation of pumps and valves during the predicted period. Reducing the time step of the optimization procedure allows a better adjustment of the system to the hydraulic changes, increasing more and more its efficiency. However, as already explained, the hydraulic models are complex, and can require a significant computational effort that can impair the real time operation (Mala-Jetmarova et al., 2017).

As the concern with the operation increased, the Supervisory Control and Data Acquisition (SCADA) of the WDNs also increased. Pressure, flow, power, tanks levels and many other parameters can be acquired, composing a database with information on the WDN behavior. Thus, data mining techniques can be used to extract useful information. Machine learning models, such as Artificial Neural Networks (ANN) have been applied with good results for leakage and pipe burst detection (Chan et al., 2018) and water demand forecasting (Ghalehkhondabi et al., 2017). In addition, ANNs can also be used as a surrogate model to estimate the WDN state (Broad et al., 2010), significantly reducing the computational effort required by the hydraulic model in real time operation.

The accuracy of the ANN is important to its effectiveness in WDNs operation, and its architecture is one of the main features to achieve this. In a Multilayer Perceptron (MLP) ANN, the definition of the number of neurons and hidden layers used is almost empirical, and can significantly change according to the problem being studied. Thus, in this paper the architecture of an MLP-ANN used as a pressure estimator will be studied, aiming establish guidelines for this problem and reduce the effort in future ANNs creation. Three different ANNs will be tested, all of them with three hidden layers, varying the number of neurons from 10 to 50. Each ANN will be trained using 12 different functions, and the results obtained will be compared using two parameters: mean and maximum pressure error.

ARTIFICIAL NEURAL NETWORKS

The last decades have been marked by the increasing of computer processing capacity and the popularization of systems' monitoring. Both facts are fundamental for understanding the boom in machine learning techniques and data mining on several fields. Among the techniques, a set of them are inspired on biological functioning of brain, the artificial neural networks (ANNs).

One of the most known and applied ANN is the multi-layer perceptron (MLP). This algorithm maps output space based on several linear processing units, the perceptron, linked one each other in layers. Each perceptron receives input data, calculate a linear combination of inputs by weights and process the result by a non-linear function. The arrange of several perceptrons on different layers makes input data be processed several times by different weighs and functions, resulting on a powerful tool for regression problems (Taud & Mas, 2018).

MLPs are submitted to a training process, also known as learning process to map with accuracy the output space. For better understand the training, let's take a familiar problem: least square method for linear functions. Considering the adjust of a linear function: f(x) = ax + b, the least squared method finds the open parameters a and b for minimize an error function. This error function is calculated, in general, using the mean squared error (MSE), written as Equation 1:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left[y_i - f\left(x_i\right)\right]^2 \quad (1)$$

where y_i is the observed value of output space, x_i is the input data linked to y_i and N is the number of observed points. In this simple case, the coefficients a and b are obtained directly from deriving the MSE in terms of a and b.

For the MLP, the weights and biases are the parameters to be adjusted and, depending on the number of neurons, layers and inputs or outputs, the number of weights and biases can arrive easily on hundreds or thousands. It is expected that with a greater number of parameters to be adjusted (hyperparameters) (i.e., greater number of hidden layers and more neurons per hidden layer) the ANN can memorize more detailed characteristics. However, the high number of parameters and training iteration can provide overfitting and/or overtraining problems and ANN accuracy becomes poorly (Tetko et al., 1997; Lin & Wu, 2021). In some ANNs where the overfitting and/or overtraining can be a problem, an additional algorithm, such as the one implemented by Antonacci et al. (2021) is necessary. Therefore, the problem of finding an optimal architecture, its weights and biases is hard and a field for application of several optimization techniques.

The hyperparameters (*Hp*) are the adjustable parameters and express the number of weights and biases to be adjusted in a given architecture for ANN, as shown in Figure 1, where x and y are the number of input and output parameters respectively, and n is the number of the neurons in the hidden layer, can be estimated according to Equation 2, where weights are the matrices $W_{ij,}$ product of two consecutive layers, being the total number of layers (j) the number of hidden layers + 2 (Input and Output) and the biases are a vector b_j for each hidden and output layer.

$$Hp = \sum_{1}^{j} \left(w_{ij} + b_{ij} \right) \quad (2)$$

Usually, the application of ANN and in the specific case of MLPs are conducted by the using of already implemented toolboxes (e.g., Deep Learning, implemented in Matlab). This toolbox provides a set of training algorithms, most of them based on first order optimization (Gradient method) or first-pseudo second order (Levemberg-Marquardt among other). Since the optimization success depend on MLP architecture (e.g., number of neuron and layers) and the generality capacity of a MLP also depends of the architecture, this work provides a set of evaluations to identify good practices on developing MLPs for the problem of state estimation in water distribution systems, specifically, the pressure estimation.

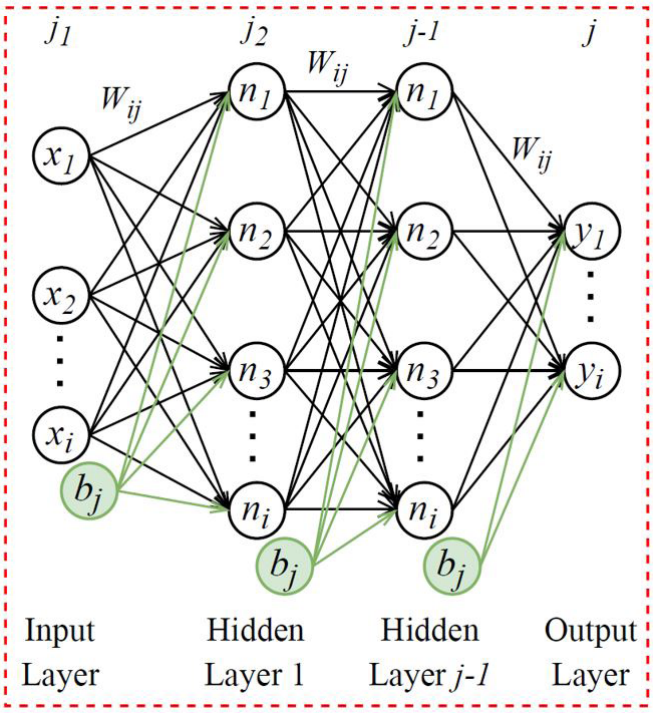

Figure 1. A given architecture for ANN.

PRESSURE ESTIMATOR MODELING

As well-known as it can be the topology of a WDN, hydraulic parameters are highly variable, both in short- and long-term periods, result for example of a pipe burst or a deterioration of pipe roughness. To develop an accurate model is hard, and can expend a high computational effort. Thus, a surrogate model to estimate these parameters can be an important tool to take rapid and efficient decisions. Meirelles et al. (2018) propose the creation of a simulated database, where the hydraulic model is used to calculate pressure in all nodes of the WDN varying nodal demand and pipe roughness randomly. Pressure values of monitored nodes are used as input to train an ANN, while the remaining values are the output of the surrogate model as shown in Figure 2. Meirelles et al. (2017) show this approach can be used for the WDN model calibration with good results, which can be then further used to optimize the operation. However, any modifications in the WDN will demand the creation of a new ANN. Another approach, proposed by Xu et al. (2020) is to use only the values of the existing pressure sensors, and compose one or more ANNs to predict the pressure on these nodes. If the estimated and measured values are different by a predefined margin, a trigger is activated, indicating an anomaly on the WDN to be further investigated.

In both cases, the accuracy of the ANN in the pressure estimation is highly important, as it will be used to support operational decisions. Thus, the following three conditions of the ANN training are relevant: i) the database composition, with a large series without redundancy and containing extreme values (minimum and maximum values); ii) the ANN architecture, with a minimum number of neurons and hidden layers necessary for an accurate ANN; and iii) the optimization training algorithm and its parameters, to achieve a fast convergence.

The performance of the trained ANN can be evaluated comparing its results with a test set. For each node, the maximum and average errors can be calculated trough Equations 3 and 4.

$$\varepsilon_{mean} = \frac{\sum_{i=0}^{t} |p_m - p_s|}{t} \quad (3)$$

$$\varepsilon_{max} = max\left(|p_{m,i} - p_{s,i}| \right) \quad (4)$$

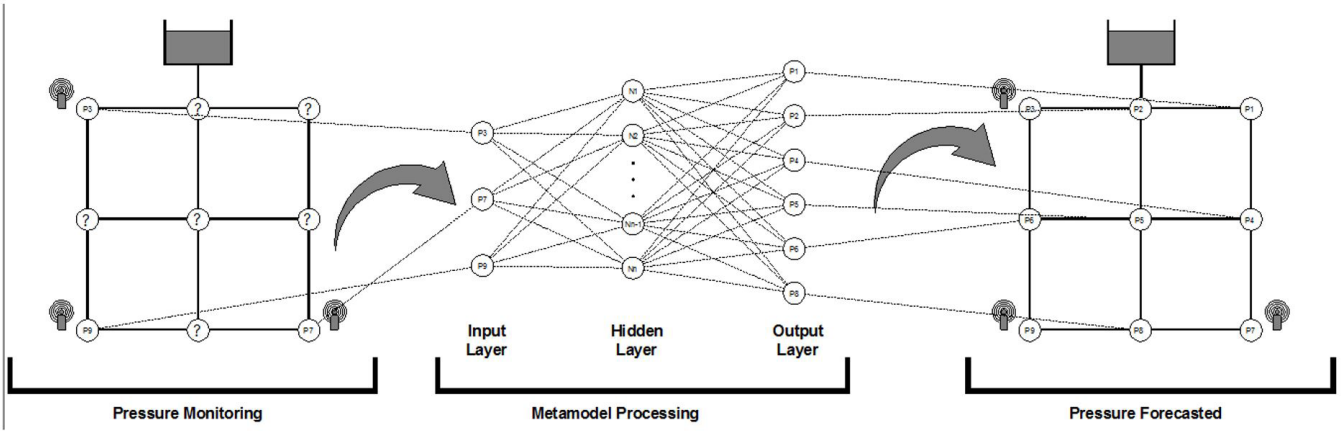

Figure 2. Example of a trained ANN results. Source: Meirelles et al. (2018).

where ε_{med} is the average error for a given node, ε_{max} is the maximum error for a given node, t is the test set size, p_m is the monitored pressure, p_s is the estimated pressure by the ANN.

CASE STUDY

The case study is the Oberlin network (OBCL-1), a model originally inspired by a part of the water distribution network that supplies the Oberlin neighborhood, located in the city of Harrisburg, Pennsylvania, northeastern of United States. The model is part of the database of a series of benchmarking networks available for research by the Task Committee on Research Databases for Water Distribution Systems of the American Society of Engineers (Hernandez et al., 2016).

The OBCL-1 network infrastructure comprises 269 nodes, 294 pipes, one pressure reducing valve (PRV) and the supply is carried out through a reservoir and a pump station immediately downstream. The monitored data refer to the pipe flow at the reservoir outlet and the pressure of 10 nodes scattered throughout the infrastructure and located as shown in Figure 3.

The database was created through simulations in the Epanet software, and represents the monitoring of 3 years of operation, with data collected every hour, totaling 26,279 readings. Thus, the database consists of pressure on all 269 nodes of the network, and the values of the 10 monitored nodes are used as input for the ANN training, in addition with the output flow of the reservoir (Figure 3). The remaining data will create the ANN output matrix. The database was randomly divided into a ratio of 85% (22,337) for the training data and the remainder, 15% (3,942), for validation of the results, and then used in the various ANN configurations. It is emphasized that this division into training and validation was the same for all tests performed in this work.

The methodology used was based on determining the best architecture of the ANN to act as pressure estimator from the observed data available with the use of the Deep Learning framework in the MATLAB© software. For this, it was used the

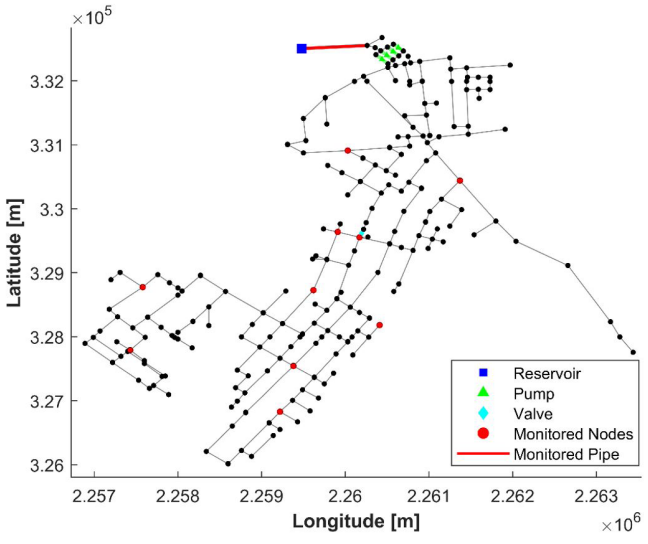

Figure 3. Oberlin Network (OBCL-1).

MLP model in which the inputs corresponds to the 11 monitored points and the outputs is the pressure in the 259 unmonitored nodes. The hidden layers of the perceptrons are varied for different arrangements, ranging from one to three, and associated with the variation in the number of neurons, corresponding to 10, 20 or 50, always equal per hidden layer, resulting on nine different arrangements with a number of hyperparameters (*Hp*) to be adjusted (weights and biases) calculated as demonstrated in Equation 2 with a degree of freedom (*DoF*) being to the difference between the training data (22,337) minus the hyperparameters (*Hp*) where the results are shown in Table 1. The training parameters adopted were: a maximum number of validation failure of 15, a maximum number of 1e^6 epochs, random division of weights, evaluation of performance through the mean square error (MSE) and other parameters adopted with the MATLAB® default values, as shown in Table 2.

In the training of the ANN, each architecture configuration is evaluated for all twelve training algorithms available in MATLAB® as presented in Table 3. The performance goal is to comply the maximum errors below the limit of 2 meters for the pressure estimation, a value considered acceptable by the Water Research Centre (1989) in calibration processes. However, other training results are evaluated as mean absolute error.

RESULTS

Tables 4, 5 and 6 show the results of the performance of the training algorithms for each architecture studied. Figure 4 compares two weeks between observed and estimated pressure, while Figure 5 details the frequency of maximum absolute error below the WRC reference of 2 m according to the number of hidden layers and neurons used. Figure 6 presents the spatial distribution of case study errors for the best architecture found and Figure 6 the frequency at which a given absolute error is observed.

It is observed that one third of the available training algorithms could not predict the pressures for the OBCL-1 case study from the available database, as their architecture were not evaluated due to the impossibility to start the iterations to training ANN, namely: Levenberg-Marquardt, BFGS quasi-Newton, Bayesian regularization and Gradient descent with momentum. Two other functions, Gradient descent with adaptive learning rate and Gradient descent with momentum and adaptive learning rate, presented maximum absolute error results well above the 2

Table 1. The number of hyperparameters to be adjusted for each architecture.

Architecture	W_{ij}	b_j	*Hp*	*DoF*
[10]	2,700	269	2,969	19,368
[20]	5,400	279	5,679	16,658
[50]	13,500	309	13,809	8,528
[10, 10]	2,800	279	3,079	19,258
[20, 20]	5,800	299	6,099	16,238
[50, 50]	16,000	359	16,359	5,978
[10, 10, 10]	2,900	289	3,189	19,148
[20, 20, 20]	6,200	319	6,519	15,818
[50, 50, 50]	18,500	409	18,909	3,428

Table 2. Training parameters adopted.

Training parameters	Values	Description
net.trainParam.epochs	$1.0e^{6}$	Maximum number of epochs to train
net.trainParam.goal	0	Performance goal
net.trainParam.min_grad	$1.0e^{-6}$	Minimum performance gradient
net.trainParam.max_fail	15	Maximum validation failures
net.trainParam.sigma	$5.0e^{-5}$	Determine change in weight for second derivative approximation
net.trainParam.lambda	$5.0e^{-7}$	Parameter for regulating the indefiniteness of the Hessian
net.trainParam.min_step	$1.0e^{-6}$	Minimum step length

Table 3. Training algorithms available in MATLAB®.

No.	Acronym	Algorithm	Description
1	SCG	trainscg	Scaled conjugate gradient backpropagation
2	LM	trainlm	Levenberg-Marquardt backpropagation
3	GDA	traingda	Gradient descent with adaptive learning rate backpropagation
4	GDX	traingdx	Gradient descent with momentum and adaptive learning rate backpropagation
5	RP	trainrp	Resilient backpropagation
6	CGF	traincgf	Conjugate gradient backpropagation with Fletcher-Reeves updates
7	CGB	traincgb	Conjugate gradient backpropagation with Powell-Beale restarts
8	BFG	trainbfg	BFGS quasi-Newton backpropagation
9	CGP	traincgp	Conjugate gradient backpropagation with Polak-Ribiére updates
10	OSS	trainoss	One-step secant backpropagation
11	BR	trainbr	Bayesian regularization backpropagation
12	GDM	traingdm	Gradient descent with momentum backpropagation

Table 4. Results for one Hidden Layer.

Training Algorithm	1 Hidden Layer					
	[10]		[20]		[50]	
	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}
1	3.4	0.02	2.6	0.02	1.9	0.01
2	-	-	-	-	-	-
3	11.8	0.49	17.1	0.66	37.1	0.84
4	35.9	0.60	9.1	0.27	21.5	0.37
5	2.0	0.03	2.1	0.02	2.0	0.02
6	4.1	0.09	4.0	0.03	3.0	0.02
7	3.6	0.02	3.3	0.02	3.1	0.03
8	-	-	-	-	-	-
9	3.9	0.06	2.9	0.04	2.9	0.02
10	5.0	0.10	3.7	0.02	2.5	0.03
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Table 5. Results for two Hidden Layer.

Training Algorithm	2 Hidden Layer					
	[10, 10]		[20, 20]		[50, 50]	
	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}
1	2.3	0.02	1.6	0.01	1.5	0.02
2	-	-	-	-	-	-
3	34.2	1.58	69.4	3.12	92.6	4.17
4	44.7	0.80	68.6	3.71	69.7	2.29
5	2.0	0.02	1.8	0.02	1.5	0.02
6	4.1	0.09	4.1	0.09	3.4	0.05
7	4.1	0.09	4.0	0.09	4.0	0.07
8	-	-	-	-	-	-
9	4.1	0.10	4.1	0.09	1.8	0.02
10	2.6	0.03	1.7	0.02	1.3	0.02
11	-	-	-	-	-	-
12	-	-	-	-	-	-

Table 6. Results for tree Hidden Layer.

Training Algorithm	3 Hidden Layer					
	[10, 10, 10]		[20, 20, 20]		[50, 50, 50]	
	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}
1	1.9	0.03	2.0	0.02	1.8	0.02
2	-	-	-	-	-	-
3	45.7	2.28	84.5	4.40	110.4	9.42
4	23.1	0.37	63.4	3.35	61.6	2.27
5	3.7	0.01	2.1	0.01	1.7	0.03
6	4.1	0.13	4.6	0.13	4.1	0.09
7	4.0	0.10	4.1	0.10	4.0	0.09
8	-	-	-	-	-	-
9	4.2	0.12	4.1	0.10	4.1	0.10
10	4.1	0.10	4.2	0.10	4.1	0.09
11	-	-	-	-	-	-
12	-	-	-	-	-	-

meters reference. The performance of these two functions could be improved increasing the maximum number of validation failures used, but to maintain the comparison criterion among the functions the value of 15 was kept.

According to Tables 4 and 6, it is worth noticing that, for one and three hidden layers, only the Scaled conjugate gradient and Resilient functions of the remaining six present absolute maximum error lower than 2 m. For two hidden layers, according to Table 5, the conjugate gradient with Polak-Ribiére updates and One-step secant functions also presented results within the desired limit totaling four of the six functions with absolute maximum error lower than 2 m. The other two functions, Conjugate gradient with Fletcher-Reeves updates and Conjugate gradient with Powell-Beale restarts presented maximum errors around 4 m for all configurations, so they are unfeasible for the case study.

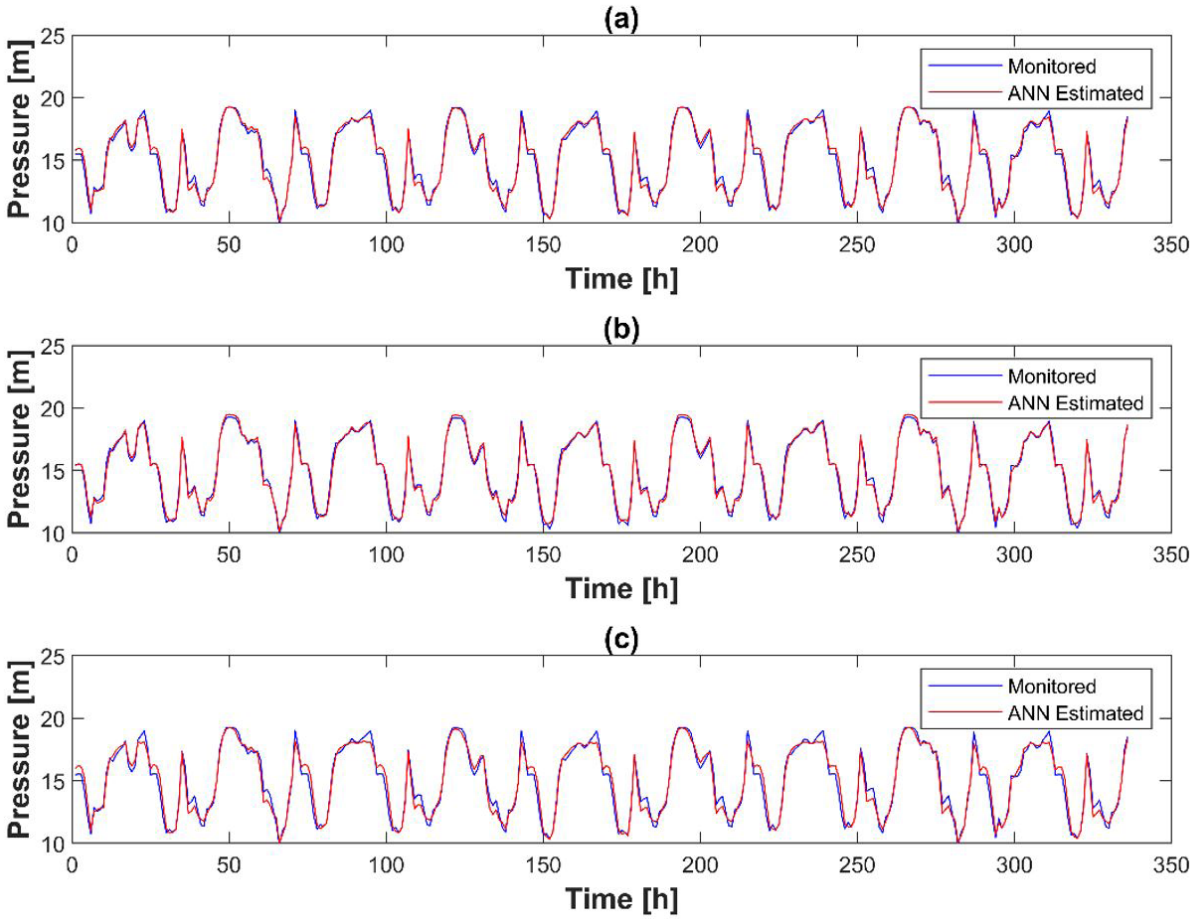

Figure 4. Comparison between monitored and estimated data to node 157 for ANN with two hidden layers and 50 neurons per hidden layer for different Training Algorithms: (a) Scaled conjugate gradient backpropagation; (b) Resilient backpropagation; (c) One-step secant backpropagation.

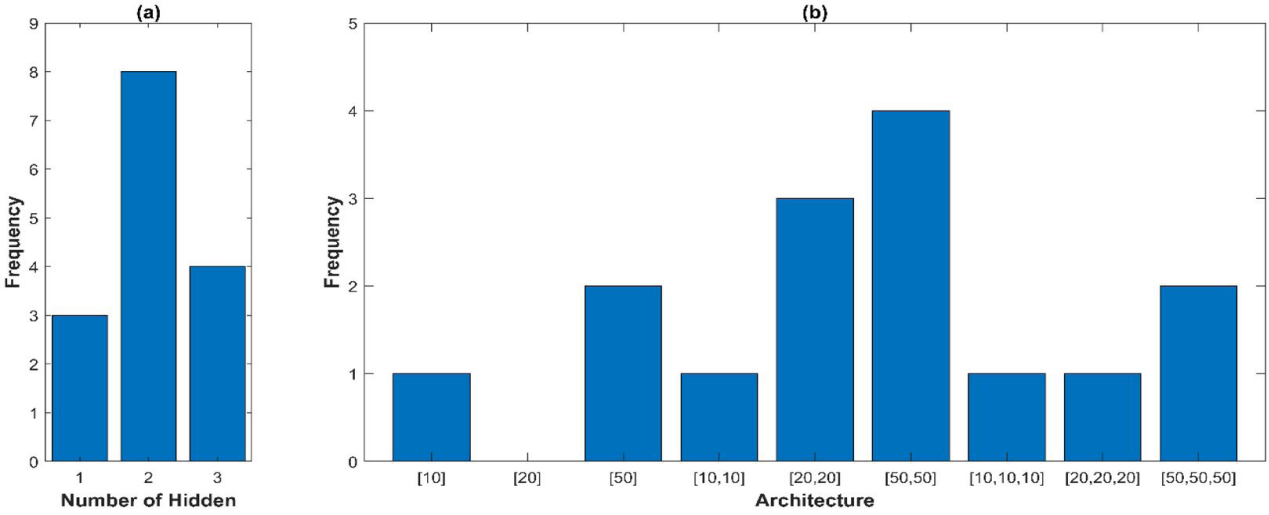

Figure 5. Observations of absolute maximum error less than 2 meters: (a) Observations per Hidden Layers; (b) Observations per architecture.

As shown in Figure 5a, the architecture with only one hidden layer is the one with the lowest frequency of errors lower than 2 m, with only three observations below this value, one with the Scaled conjugate gradient function with 50 neurons and the other two for resilient function with 10 and 50 neurons, respectively. The architecture with two hidden layers presents the highest frequency of maximum errors lower than 2 m, with eight cases in total. In addition, it was also the one that presented the lowest maximum

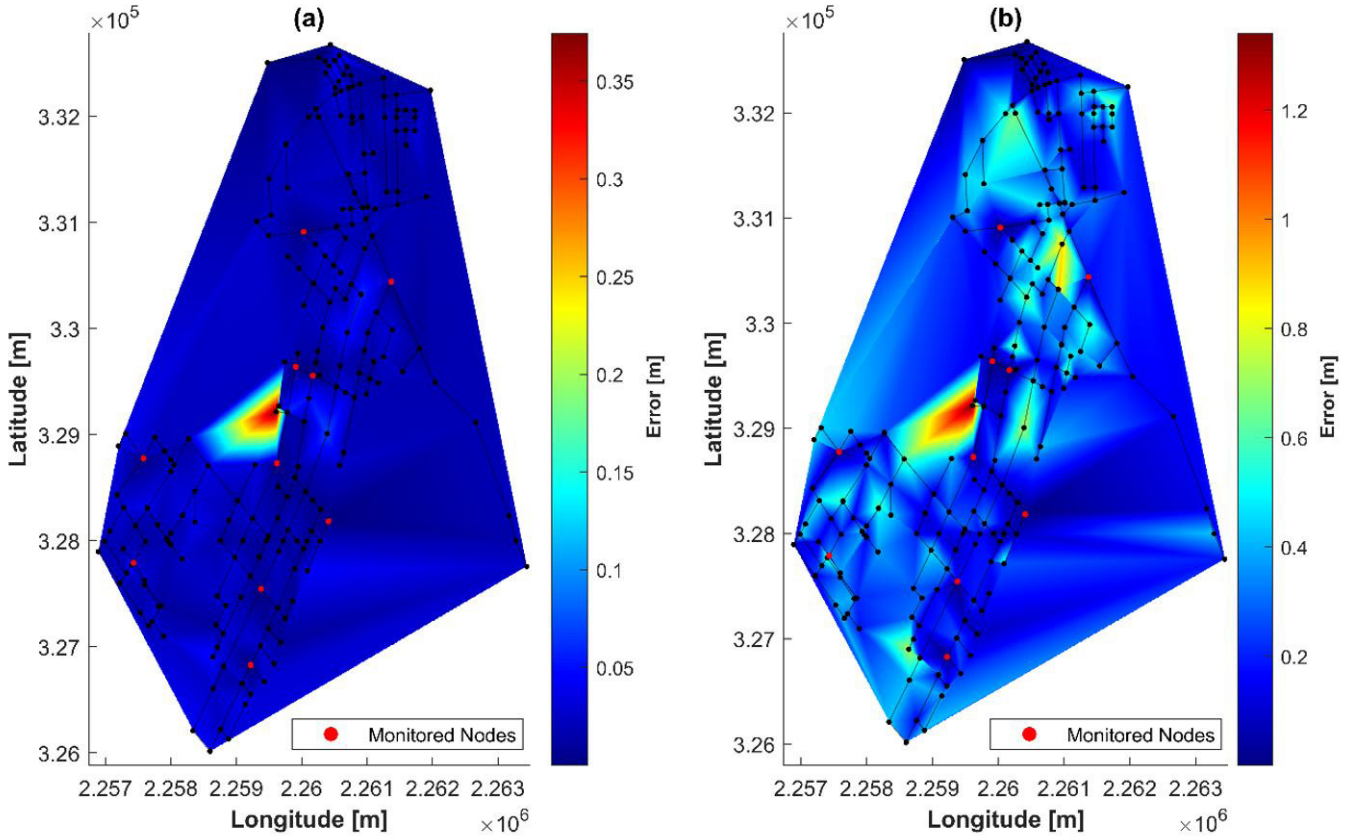

Figure 6. Spatial distribution of absolute error for 2 Hidden Layers, 50 neurons per Hidden Layer and one-step secant training algorithm: (a) Average absolute error; (b) Maximum absolute error.

absolute error, of 1.3 meters for the One-step secant function with 50 neurons in each layer. Finally, the architecture with three hidden layers presented results similar to the one-layer case, with only four tests with maximum absolute errors lower than 2 m, observed when using the Scaled conjugate gradient functions for 10, 20 and 50 neurons and Resilient for 50 neurons.

Also, according to Figure 5b, it is observed that, for the different number of hidden layers, when 50 neurons were used, the frequency of absolute maximum error below 2 meters increased. In agreement with the results showed in Figure 5a, the two hidden layers architecture had the best results, with four tests below the error reference, indicating a predominance of optimized architecture for the case study. Even with a more complex architecture for 3 hidden layers (i.e., [50,50,50]) and the ability to memorize more detailed characteristics, the results performed were worse than for 2 hidden layers (i.e., [50,50)]), being justified by the stopping criterion less than the maximum number of iterations and by the overfitting and/or overtraining problems as highlighted by Tetko et al. (1997) and Lin & Wu (2021) and caused by the lowest degree of freedom (*DoF*). It is also important to highlight that, considering different training algorithms, optimal parameters can be achieved for different architectures and different number of iterations, which also justifies simple architectures (i.e., [20,20]) with better performance than more complex architectures (i.e., [20,20,20]) with different training algorithms.

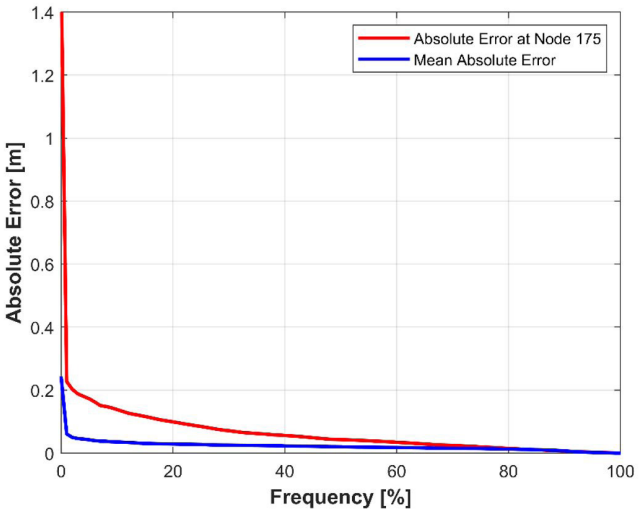

Figure 7. Frequency in observing errors.

Figure 6 shows the spatial distribution of the mean absolute error and maximum absolute error in each of the 259 unmonitored nodes and Figure 7 the frequency of a given absolute error. It is observed that, in less than 10% of the estimated pressure period the absolute error is higher than 0.8 m in node 175, while on average, the absolute error is higher than 0.2 m only in 1%.

Thus, Figure 6 and Figure 7 justify the high values for maximum absolute error and lower values for the mean absolute error presented in Tables 3, 4 and 5, since the concentration of the error occurs almost exclusively in node 175. So, it is a restricted problem since for the other 258 nodes the ANN can estimate pressure with excellent results.

CONCLUSIONS

There are several possibilities of architectures for ANN and different training algorithms, and, finding the ideal arrangement, especially that respects the restriction(s) of the problem studied, is not an easy task. Thus, this work aimed to contribute to the evaluation of different architectures and training algorithms of ANN to estimate pressures. It is noteworthy that different training parameters and architectures with higher numbers of hidden layers and/or neurons per hidden layers may present results that meet the restriction of the problem, however, it depends on a higher computational effort in the training of ANN, and problems with overfitting and overtraining can increase, and it may be necessary to implement some additional technique to avoid it, but not discussed in the present paper. However, higher numbers of hidden layers and neurons per hidden layers or a specific algorithm to avoid overfitting have not been necessary for the case study, since the results found satisfied the system restriction.

Although there were 12 training algorithms available at MATLAB®, four were not even able to start training the ANN, and of the remaining eight, only half achieved results that respected the restrictions of the problem. For one and three hidden layers, only the Scaled conjugate gradient and Resilient functions showed errors in pressure estimation below 2 meters. When using two hidden layers, the frequency of errors below 2 meters was the highest. In addition, the configuration with 50 neurons presented the smallest errors, reaching the lowest maximum absolute error of 1.3 meters with the one-step secant training algorithm. Thus, the predominance of optimized architecture for the case study with 2 layers and 50 neurons in each layer was observed.

DATA AVAILABILITY STATEMENT

Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request, like as:

- Database;
- Epanet model;
- Adjusted parameters (weight and bias trained values).

The ANNs created in this study are specific to the WDN and the sensors placement used. For any other WDN, and also, for a different sensor placement in this exactly WDN, a new ANN have to be trained.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

Antonacci, Y., Minati, L., Faes, L., Pernice, R., Nollo, G., Toppi, J., Pietrabissa, A., & Astolfi, L. (2021). Estimation of Granger causality through Artificial Neural Networks: applications to physiological systems and chaotic electronic oscillators. *PeerJ. Computer Science*, *7*, e429. PMid:34084917. http://dx.doi.org/10.7717/peerj-cs.429.

Brentan, B., Meirelles, G., Luvizotto Junior, E., & Izquierdo, J. (2018). Joint operation of pressure-reducing valves and pumps for improving the efficiency of water distribution systems. *Journal of Water Resources Planning and Management*, *144*(9), 04018055. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000974.

Broad, D. R., Maier, H. R., & Dandy, G. C. (2010). Optimal operation of complex water distribution systems using metamodels. *Journal of Water Resources Planning and Management*, *136*(4), 433-443. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000052.

Chan, T. K., Chin, C. S., & Zhong, X. (2018). Review of current technologies and proposed intelligent methodologies for water distributed network leakage detection. *IEEE Access: Practical Innovations, Open Solutions*, *6*, 78846-78867. http://dx.doi.org/10.1109/ACCESS.2018.2885444.

Fecarotta, O., Ramos, H. M., Derakhshan, S., Del Giudice, G., & Carravetta, A. (2018). Fine tuning a PAT hydropower plant in a water supply network to improve system effectiveness. *Journal of Water Resources Planning and Management*, *144*(8), 04018038. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000961.

Fontana, N., Giugni, M., Glielmo, L., Marini, G., & Zollo, R. (2018). Real-time control of pressure for leakage reduction in water distribution network: field experiments. *Journal of Water Resources Planning and Management*, *144*(3), 04017096. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000887.

Ghalehkhondabi, I., Ardjmand, E., Young 2nd, W. A., & Weckman, G. R. (2017). Water demand forecasting: review of soft computing methods. *Environmental Monitoring and Assessment*, *189*(7), 313. PMid:28585040. http://dx.doi.org/10.1007/s10661-017-6030-3.

Hernandez, E., Hoagland, S., & Ormsbee, L. E. (2016). *WDSRD: A Database for Research Applications*. Retrieved in 2020, December 28, from http://www.uky.edu/WDST/database.html.

Lin, C. J., & Wu, N. J. (2021). An ANN model for predicting the compressive strength of concrete. *Applied Sciences*, *11*(9), 3798. http://dx.doi.org/10.3390/app11093798.

Mala-Jetmarova, H., Sultanova, N., & Savic, D. (2017). Lost in optimisation of water distribution systems? A literature review of system operation. *Environmental Modelling & Software*, *93*, 209-254. http://dx.doi.org/10.1016/j.envsoft.2017.02.009.

Meirelles, G., Brentan, B. M., Manzi, D., & Luvizotto Junior, E. (2018). Metamodel for nodal pressure estimation at near real-time in water distribution systems using artificial neural networks. *Journal*

of Hydroinformatics, *20*(2), 486-496. http://dx.doi.org/10.2166/hydro.2017.036.

Meirelles, G., Manzi, D., Brentan, B., Goulart, T., & Luvizotto Junior, E. (2017). Calibration model for water distribution network using pressures estimated by artificial neural networks. *Water Resources Management*, *31*(13), 4339-4351. http://dx.doi.org/10.1007/s11269-017-1750-2.

Salomons, E., & Housh, M. (2020). A practical optimization scheme for real-time operation of water distribution systems. *Journal of Water Resources Planning and Management*, *146*(4), 04020016. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0001188.

Taud, H., & Mas, J. F. (2018). Multilayer perceptron (MLP). In M. T. C. Olmedo, M. Paegelow, J.-F. Mas & F. Escobar (Eds.), *Geomatic approaches for modeling land change scenarios* (pp. 451-455). Cham: Springer. http://dx.doi.org/10.1007/978-3-319-60801-3_27.

Tetko, I. V., Villa, A. E., & Tetko, I. V. (1997). An enhancement of generalization ability in cascade correlation algorithm by avoidance of overfitting/overtraining problem. *Neural Processing Letters*, *6*(1), 43-50. http://dx.doi.org/10.1023/A:1009610808553.

Water Research Centre – WRC. (1989). *Network analysis: a code for practice*. Swindon: WRC.

Xu, Q., Chen, Q., Qi, S., & Cai, D. (2015). Improving water and energy metabolism efficiency in urban water supply system through pressure stabilization by optimal operation on water tanks. *Ecological Informatics*, *26*, 111-116. http://dx.doi.org/10.1016/j.ecoinf.2014.09.007.

Xu, Z., Ying, Z., Li, Y., He, B., & Chen, Y. (2020). Pressure prediction and abnormal working conditions detection of water supply network based on LSTM. *Water Supply*, *20*(3), 963-974. http://dx.doi.org/10.2166/ws.2020.013.

Authors contributions

Rui Gabriel Modesto de Souza: Performed the methodology, obtained the results and wrote the text.

Bruno Melo Brentan: Performed the methodology and revised the results and text.

Gustavo Meirelles Lima: Contributed with technical notes and revised the text.

Editor-in-Chief: Adilson Pinheiro

Associated Editor: Carlos Henrique Ribeiro Lima