A method for measuring the robustness of diagnostic models for predicting the break size during LOCA

Xiange Tian[1], Victor Becerra[2], Nils Bausch[3], Gopika Vinod[4], and T.V. Santhosh[5]

[1, 2, 3]*School of Engineering, University of Portsmouth, Portsmouth, PO1 2UP, UK*

xiange.tian@port.ac.uk
victor.becerra@port.ac.uk
nils.bausch@port.ac.uk

[4, 5]*Reactor Safety Division, Bhabha Atomic Research Centre, India*

vgopika@barc.gov.in
santutv@barc.gov.in

ABSTRACT

The diagnosis of loss of coolant accidents (LOCA) in nuclear reactors has attracted a great deal of attention in condition monitoring of nuclear power plants (NPPs) because the health of cooling system is crucial to the stability of the nuclear reactor. Multi-layer perceptron (MLP) neural networks have commonly been applied to LOCA diagnosis. The data used for training these models consists of a number of time-series data sets, each for a different break size, with the transient behavior of different measurable variables in the coolant system of the reactor following a LOCA. It is important to select a suitable architecture for the neural network that delivers robust results, in that the predicted break size is deemed to be accurate even for a break size that is not included in the training data sets. The objective of this paper is to present a simple method for measuring the robustness of diagnostic models for predicting the break size during the loss of coolant accidents. A robustness metric is proposed based on the leave-one-out approach and the mean squared error resulting from a diagnostics model. Using this metric it becomes possible to compare the robustness of different diagnostic models. Given data obtained from a high fidelity simulation of the coolant system of a nuclear reactor, four different diagnostic models are obtained and their properties compared and discussed. These models include a fully connected multi-layer perceptron with one hidden layer, a fully connected multi-layer perceptron with two hidden layers, a multi-layer perceptron with one hidden layer that is pruned using the optimal brain surgeon algorithm, a group method of data handling (GMDH) neural network, and an adaptive network based fuzzy inference system (ANFIS).

1. INTRODUCTION

The reactor coolant system is the key part of NPPs. LOCA can result in severe consequences for the plant, environment, personnel, and the public in the area around the plant. Therefore, to detect and diagnose LOCA, a great deal of attention has been paid to the monitoring of coolant system. Depending on the severity of the LOCA, it could be necessary to take fast and effective actions to protect the surrounding environment and public around the site. The diagnosis of LOCA in NPPs is often considered as a transient identification problem. Artificial neural networks (ANNs) are employed in most NPP transient identification studies, such as MLP neural network (Moshkbar-Bakhshayesh & Ghofrani, 2013), GMDH network (Lee, No, Na, Ahn, & Park, 2011), and neuro-fuzzy system (da Costa, Mol, de Carvalho, & Lapa, 2011). The classical MLP neural network has difficulty with generalization when the training data is limited (Hassibi, 1993 & Norgaard et al., 2000). GMDH is a kind of growing network which is able to optimize its structure. Neuro-fuzzy neural networks combine neural network type architectures with fuzzy logic. This type of network has been applied to condition monitoring systems because of its fast learning, on-line adaptability, and low computational requirements. It is important to select an optimal architecture from many choices for the neural network that delivers robust results. To achieve this objective, different attempts have been made to automate the architecture selection. One common strategy is to start with a fully connected network architecture which, in principle, is large enough to describe the system, then the weights are eliminated one at a time until the optimal architecture has been reached, e.g. optimal brain surgeon algorithm (OBS). Another strategy stars with a small network architecture and then gradually increase it, e.g. GMDH network.

To compare the robustness of diagnostic models for predicting the break size during LOCA, this paper presents a robustness metric inspired by the leave-one-out approach.

2. METHODOLOGY

This section introduces the working principles of the neural networks investigated in this paper.

2.1. Multi-layer perceptron

MLP is a kind of feed-forward artificial neural network where a large number of processing elements are interconnected in a directed graph to create a functional mapping from the input data space to the output target space after training. A basic MLP contains three layers (input layer, hidden layer, and output layer) as shown in Figure 1. With the exception of the input layer, all nodes in other layers contain either linear or non-linear activation functions.

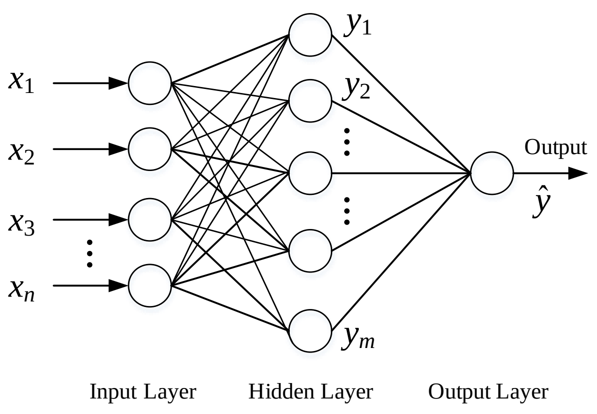

Figure 1. One hidden layer MLP

MLP is normally trained through the widely used backpropagation algorithm. In the backpropagation algorithm, the network is provided with inputs and corresponding target outputs. All the synaptic weights of the network are initialized randomly and adjusted in order to minimize a certain objective function, which depends on estimation error in this case. The input-output equation of a neuron in the hidden layer is [37]:

$$y_k = \Phi_k\left(\sum_{j=1}^{n} w_{kj}x_j + \theta_k\right) \tag{1}$$

where Φ_k is the activation function of the hidden neuron and is normally taken as a non-linear function, such as a sigmoid function, y_k is the output of the *k*th hidden neuron, θ_k is the bias value of the *k*th hidden neuron, and w_{kj} is the synaptic weight value from input x_j to the hidden neuron k. The output of the neural network is given by:

$$\hat{y} = \Phi_o\left(\sum_{k=1}^{m} w_{ok}y_k + \theta_o\right) \tag{2}$$

where Φ_o is the activation function of the output neuron and is normally taken as a linear function, $\hat{y}$ is the final output of the MLP network, θ_o is the bias value of the output neuron o, and w_{ok} is the synaptic weight value from the hidden neuron k to the output neuron o.

In this paper, the Levenberg-Marquardt algorithm is used for training the neural network because it is the fastest method for training moderate-sized feed-forward neural networks (up to several hundred weights). The application of Levenberg-Marquardt to neural network training is described in (Hagan & Menhaj, 1994). According to the Levenberg-Marquardt algorithm, the neural network training weights are adjusted with:

$$w(n+1) = w(n) - (\mathbf{J}^{\mathrm{T}}(n)\mathbf{J}(n) + \mu\mathbf{I})^{-1}\mathbf{J}^{\mathrm{T}}(n)e(n) \tag{3}$$

Here, $\mathbf{J}$ is Jacobian matrix, which defined as Eq. (4), $e(n)$ is the error between the output and target, and μ is the step size.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_m} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_n}{\partial w_1} & \frac{\partial e_n}{\partial w_2} & \cdots & \frac{\partial e_n}{\partial w_m} \end{bmatrix} \tag{4}$$

2.2. Optimal brain surgeon algorithm

In neural networks, the regularization problem is often cast as minimizing the number of connection weights. Without such weight elimination overfitting problems and thus poor generalization will result. Conversely, if there are too few weights, the network might not be able to learn the training data. For fully connected networks, the architecture selection problem is reduced to choosing a number of hidden units. The simplest procedure for determination of an adequate number of hidden units is to increase their number gradually while evaluating the test error. When a number of hidden units has been reached above which the gain in generalization is insignificant, the network is accepted. If the training set is very limited, it is important that the network architecture is chosen wisely in that it should contain only the most essential weights. The architecture selection is in this case much harder since it will also be difficult to set aside a data set for test purposes. The optimal brain surgeon procedure proposed by Hassibi et al. (Hassibi, 1993 & Norgaard et al., 2000) can be described by following steps:

1. Train a reasonably large network to minimum error

2. Compute the Hessian matrix and invert it, $\mathbf{H}^{-1}$

3. Find the q that gives the smallest saliency $L_p = w_q^2 / (2[\mathbf{H}^{-1}]_{qq})$. If this candidate error increase is lower than the error function used in training process, then the qth weight should be deleted, and continue with step 4 of this algorithm; otherwise go to step 5.

4. Use the q from step 3 to update all weights using $\delta w = w_q \mathbf{H}^{-1} e_q / [\mathbf{H}^{-1}]_{qq}$. Go to step 2.

5. No more weights can be deleted without large increase in the error function of training process. At this point it may be desirable to retrain the network.

2.3. Group method of data handling neural network

GMDH (Lu & Upadhyaya, 2005) takes advantage of the self-organized structure to generate a detailed system model in a systematic manner. This overcomes the tedious work of network construction, and allows focusing on the organization of effective model inputs based on physical and statistical considerations.

GMDH algorithms consider various component subsets of the base function called partial models. Coefficients of these models are estimated by the least squares method. GMDH algorithms gradually increase the number of partial model components and find a model structure with optimal complexity.

The algorithm is based on a multilayer structure using the general form, which is referred to as the Kolmogorov-Gabor polynomial (Volterra functional series).

$$\begin{aligned} y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j \\ + \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{m} a_{ijk} x_i x_j x_k \cdots \end{aligned} \tag{5}$$

where the external input vector is represented by $X = (x_1, x_2, \ldots)$, y is the corresponding output value, and a is the vector of weights and coefficients. The polynomial equation represents a full mathematical description. The whole system of equations can be represented using a matrix form as shown below:

$$y = f(X) \tag{6}$$

where

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N1} \end{bmatrix} \tag{7}$$

It can be replaced by a system of partial polynomial for the sake of simplicity as shown below:

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \tag{8}$$

where $i, j = 1, 2, \ldots, M; i \neq j$.

2.4. ANFIS neuro-fuzzy network

A neuro-fuzzy network is a fuzzy inference system equipped with a training algorithm (Jang, 1993). Since the fuzzy inference system is constructed based on fuzzy if-then rules, linguistic information can be directly incorporated and, on the other hand, numerical information is incorporated by training the fuzzy inference system to match the input-output pairs. Therefore, the fuzzy neural network combines linguistic and numerical information (mainly input-output pairs). The main advantages of the fuzzy inference system are the possibility of implementing rule of thumb, experience, intuition, and heuristics.

In this paper, a neuro-fuzzy network is designed using the ANFIS (adaptive neuro-fuzzy inference system) function in the Matlab Fuzzy Logic Toobox (MathWorks, 2016). It uses a given input/output data set to construct a fuzzy inference system (FIS), whose membership function parameters are tuned (adjusted) using either a backpropagation algorithm alone or in combination with a least squares method.

ANFIS supports the Takagi–Sugeno based systems (Takagi & Sugeno, 1985). The structure of the adaptive network is composed of five network layers i.e. layer 1 to layer 5 (with nodes and connections) as shown in Fig. 2.

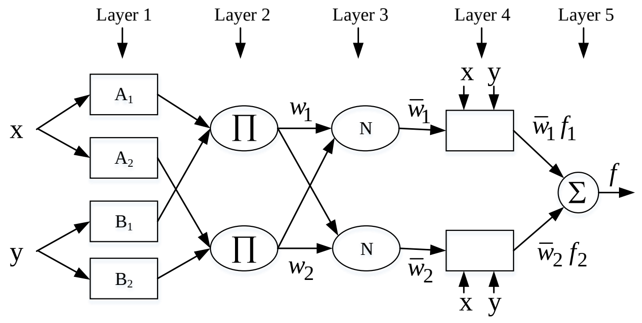

Figure 2. Architecture of a first order two rule Takagi–Sugeno type ANFIS

Assuming that the system is defined to have two inputs x_1 and x_2, one output z and fuzzy set A_1, A_2, B_1, B_2; then for a first order Takagi–Sugeno fuzzy model, having two IF-THEN rules in the common rule set, can be written using the following Eqs. (9) and (10) (Celikyilmaz & Türksen, 2009).

$$\begin{aligned}&Rule\ 1:\\ &If\ x_1\ is\ A_1\ and\ x_2\ is\ B,\\ &then\ f_1 = p_1x_1 + q_1x_2 + r_1.\end{aligned} \tag{9}$$

$$\begin{aligned}&Rule\ 2:\\ &If\ x_1\ is\ A_2\ and\ x_2\ is\ B,\\ &then\ f_2 = p_2x_1 + q_2x_2 + r_2.\end{aligned} \tag{10}$$

The neural network structure contains 5 layers excluding the input layer (Layer 0):

(1) Layer 0: input layer, has n nodes where n is number of inputs to the system.

(2) Layer 1: This layer is called as the fuzzification layer. Here the crisp input signal is fed to the node i which is associated with a linguistic label A_i or B_{i-2}. Thus, the membership function $O_{1,i}(X)$ determines the membership level (full, none or partial) of the given input. The output of each node is calculated using Eqs. (11) and (12). $O_{1,i}(X)$ is the generalized Gaussian shaped membership function used in our model development.

$$O_{1,i} = \mu_{A_i}(x_1)\quad for\ i = 1,2 \tag{11}$$

$$O_{1,i} = \mu_{B_{i-2}}(x_2)\quad for\ i = 3,4 \tag{12}$$

(3) Layer 2: The nodes in this layer are fixed and labeled as $O_{2,i}(X)$. The output of each node is the product of all the incoming signals as in the Eq. (13).

$$O_{2,i} = w_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2)\quad for\ i = 1,2 \tag{13}$$

The output of each node represents the firing strength of a rule. Also, known as the membership layer, it acts on the input variables from layer 1 as membership functions to represent them in their fuzzy sets.

(4) Layer 3: Each node in this layer calculates the ratio of the individual rule's firing strength to the sum of all rules firing strengths as in the Eq. (14). $\overline{w}_i$ represents the normalized firing strength. Hence, this layer is also known as the rule layer.

$$O_{3,i} = \overline{w}_i = w_i / (w_1 + w_2)\quad for\ i = 1,2 \tag{14}$$

Since each node in this layer calculates the normalized weights, the output signal can be thought of as the normalized firing strength of a given rule.

(5) Layer 4: This layer known as the defuzzification layer. It calculates the individual output values y from the inferring of rules from the rule base. Individual nodes of this layer are connected to the respective normalization node in layer 3 and also receive the input signal. Each node of this layer is adaptive in nature with the node function given by the Eq. (15) where p_i, q_i, r_i is a set of consequent parameters of rule i.

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i(p_ix_1 + q_ix_2 + r_i) \tag{15}$$

(6) Layer 5: This layer is known as the output layer. It has only one node and it calculates the sum of all the outputs coming from the nodes of the defuzzification layer to produce the overall ANFIS output as in Eq. (16).

$$\text{Overall output: } O_{5,i} = \sum \overline{w}_i f_i = \sum_i w_i f_i / \sum_i w_i \tag{16}$$

This architecture of this adaptive network is used to develop the ANFIS model for the prediction of the LOCA break size.

3. DATA DESCRIPTION AND PROCESSING

3.1. Parameters for training neural network

The architectures of the different neural networks considered in this study are described in Table 1.

Table 1. List of process parameters for LOCA analysis

Architecture	Parameters
1 hidden layer MLP	15 neurons in the hidden layer
Pruned 1 hidden layer MLP	Minimum number of weights: 50, 200, 400
2 hidden layer MLP **(Santhosh et al., 2011)**	19 neurons in the first hidden layer; 26 neurons in the second hidden layer
GMDH	Maximum number of inputs for neurons: 3 Degree of polynomials in neurons: 3 Maximum number of neurons in a layer: 7
ANFIS	Number of clusters is "auto"

3.2. Data description

The data for comparison of the models has been referred from the paper by Santosh *et al*, 2009 on simulation of the LOCA scenarios in NPPs.

There are 35 analog and 2 digital variables for identification of LOCA scenarios. The analog variables have time-dependent transient data whereas the digital variables indicate the status of certain process states. This type of problem is modeled as a regression problem where a set of input values is mapped to a particular break size. The data used to train the neural networks consists of multiple sets of time-series, each set corresponding to a break size, and each set containing the individual time-series of the different measured variables. For this purpose, the time scale of the transient duration is divided into several intervals such that the transient can be identified as quickly and accurately as possible in the earlier phase of its development so that the negative consequences of the fault can be minimized. Moreover, the time scale is chosen such that the number of data points at a later period of the transient is limited to reduce the computational burden when training the neural network models. A 60s transient duration is chosen in this case under the assumption that this time duration is sufficient to identify the large break LOCA event. The time scale for a transient period of 60s has appropriately been used with respect to event progression. Break sizes ranging from 20% to 200% (including 20%, 60%, 100%, 120% and 200%) have been modeled using five different network architectures.

The thermal-hydraulic data has been generated for the LOCA event occurring in reactor inlet header (RIH) with the availability of the emergency core cooling system (ECCS).

3.3. Data processing procedure

The data processing procedure is presented in Figure 3. The procedure runs 50 times to reduce the randomness induced by the initialization of weights and the split of the data sets into training, validation and test data sets. The data is divided into three groups, 70% for training, 15% for validation and 15% for testing. Only the testing results are employed for performance analysis (this is unseen data in the training process).

3.3.1. Leave-one-out method

To validate the prediction performance of neural networks to untrained break sizes, a method inspired by leave-one-out validation method is performed by taking one break size out from the training data and then including it in the testing data. The different leave-one-out tests are listed in Table 2.

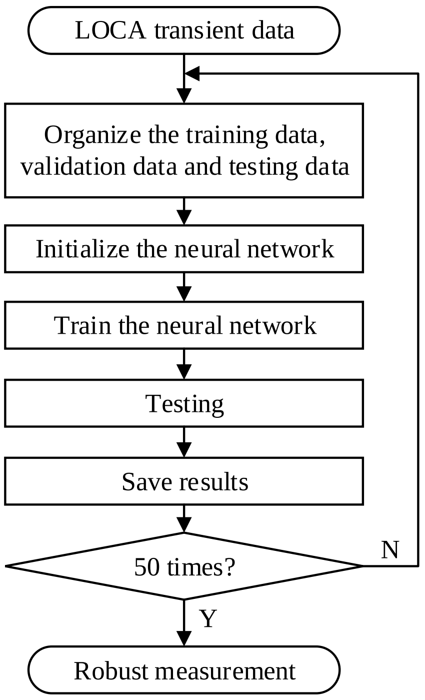

Figure 3. Data processing flowchart

Table 2. List of leave-one-out tests

Test	Training sets
1	Normal
2	Leave out 20% break
3	Leave out 60% break
4	Leave out 100% break
5	Leave out 120% break
6	Leave out 200% break

3.3.2. Robustness measurement

The prediction results of the different models studied are presented in terms of mean square error (MSE) of prediction, which is defined as follows:

$$E_i = \frac{1}{M}\sum_{j=1}^{M} e_{i,j}^2 \tag{15}$$

where e is the error between the neural network output and target, i and j are the number of test and sample, respectively, M is the total number of samples.

The following robustness measure R_m is proposed:

$$R_m = \max_{i \in [2,6]} E_i \tag{16}$$

where, E_i is the MSE for each leave-one-out case. The max function is selected to highlight the worst-case performance out of all the leave-one-out cases considered.

As previously mentioned, considering the randomness of training due to the random initialisation of weights, each network architecture was trained and tested 50 times independently. Therefore, there are 50 slightly different robustness measure values for each architecture. The overall performance of each network architecture is therefore represented by the mean value and standard deviation value of the 50 robustness measure values. The mean value of the robustness measure is calculated by

$$\bar{R}_m = \frac{1}{N} \sum_{k=1}^{N} R_m(k), N = 50 \tag{17}$$

and the standard deviation of the robustness measure is calculated by

$$\sigma = \sqrt{\frac{1}{N-1} \left| \sum_{k=1}^{N} R_m(k) - \bar{R}_m \right|^2} \tag{18}$$

A lower value of $\bar{R}_m$ indicates better robustness of the network predictions, while a lower value of σ means that the prediction performance is more consistent.

4. RESULTS AND DISCUSSION

Figures 4-8 presents the testing outputs and targets for the 6 tests listed in Table 2 from the five kinds of networks, separately. The vertical axes in the figures denote the break sizes. The outputs for the different leave-one-out cases for one given architecture are presented in each Figure 4 to 8. Note that the results shown in Figures 4-8 correspond to one particular training instance out of the 50 that were performed for each architecture and the MSE value for each sub-plot corresponds to the average result of the 50 training instances for each leave-one-out case.

Consider Figure 4 as an example, the subplot entitled “All cases” corresponds to the Test 1 in Table 2, which means the data for all break sizes are included in the training data. The subplot entitled “Without 20%” corresponds to Test 2 in Table 2, which means the data for the 20% break size is excluded from the training data. In Figure 4 it can be seen that there are large peaks in the outputs compared to the targets. From the five leave-one-out tests, it can be seen that the difference between outputs and targets for the break sizes left out during training are larger than the differences between outputs and targets when a particular break size is present in the training set.

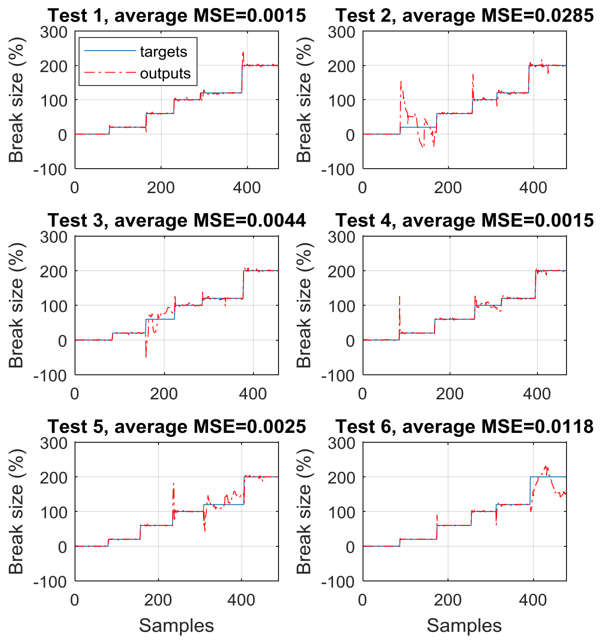

Figure 4. Testing results for the 2 HL MLP

Figure 5 and Figure 6 show the testing results for the fully connected 1 hidden layer MLP and pruned 1 hidden layer MLP, respectively. Similar with Figure 4, the average MSE value of 50 training instances for each test is shown in corresponding title. Comparing these two figures shows that the MSE values for pruned networks are smaller those of the corresponding fully connected networks, which means that the pruning method can improve the performance of neural network.

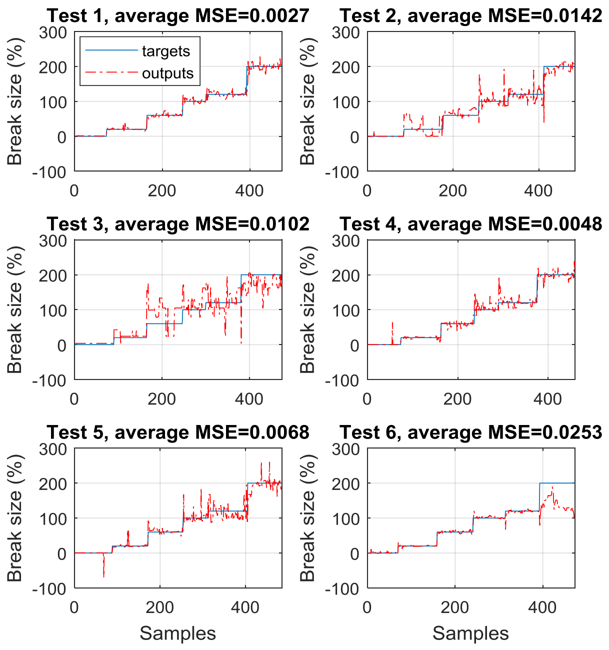

Figure 5. Testing results for the fully connected 1 HL MLP

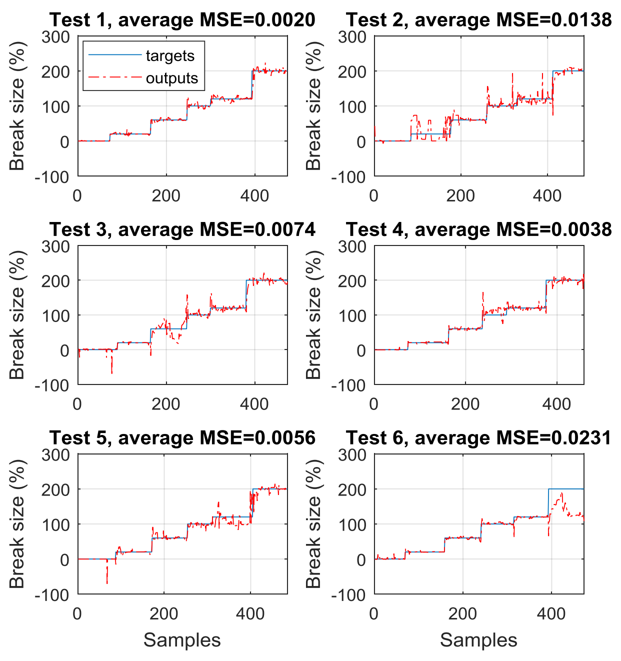

Figure 6. Testing results for the pruned 1 HL MLP

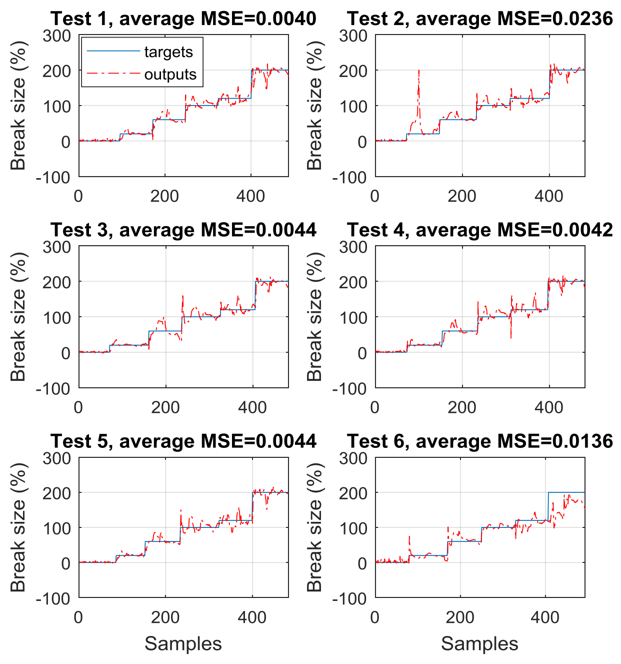

Figure 7. Testing results for the GMDH network

Figure 7 presents the testing results for the GMDH network. Comparing with the results of Figures 4-6, the MSE value of "All case" is larger, which means the prediction accuracy of GMDH is worse than the other three MLP neural networks. However, the MSE for "Without 200%" break size lower compared to the corresponding values shown in Figure 5 and 6.

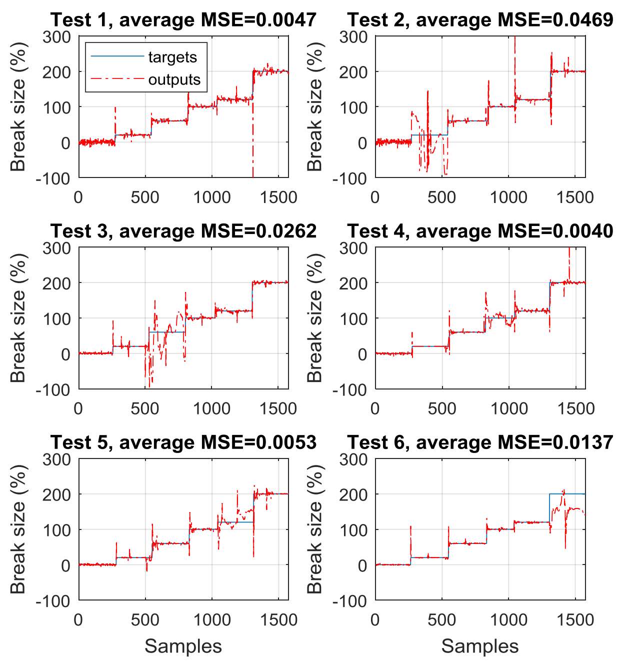

Figure 8. Testing results for the ANFIS network

Figure 8 shows the results from the ANFIS network. Comparing the MSE values with those shown in Figures 5-7, the performance of "Without 20%" and "Without 60%" are significantly worse while the performance of other tests are similar.

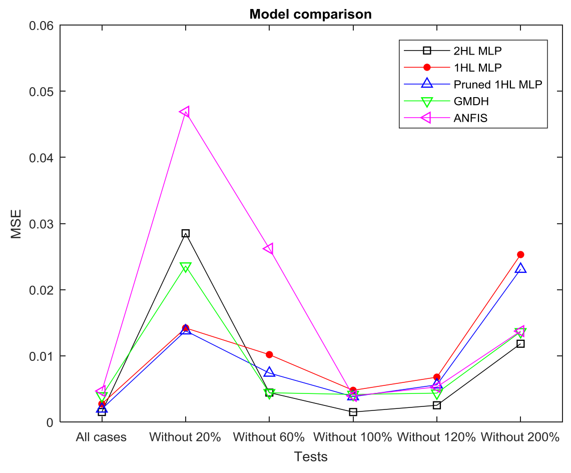

Figure 9. Comparison of models based on the average MSE

A comparison of the five different architectures based on the average MSE value for each leave-one-out case is shown in Figure 9. Note that the MSE values given for each case correspond to the average value of the 50 training instances that were performed for each architecture. From Figure 9 it can be seen that the pruned 1 hidden layer MLP has smaller MSE values than the fully connected 1 hidden layer MLP. For leaving out small break sizes, 1 hidden layer MLP architecture has better performance. For larger break sizes, 2 hidden layer architecture outperforms the other architectures.

The robustness measures obtained from the different architectures investigated in this paper are shown in Table 3. The GMDH approach has the smallest $\bar{R}_m$ value, which means this architecture has the best robustness for the untrained break size. However, its σ value is slightly larger than fully connected 1 hidden layer MLP and pruned 1 hidden layer MLP. Both $\bar{R}_m$ and σ values for the pruned 1 hidden layer MLP are smaller than the corresponding fully connected network, which means the OBS pruning algorithm was able to improve the robustness and consistency of neural network.

The $\bar{R}_m$ and σ values of the fully connected 2 hidden layer MLP and ANFIS are higher than $\bar{R}_m$ and σ values of the other three network architectures. This indicates that these two architectures have worse robustness and consistency than the other architectures considered here.

Table 3. Robustness comparison of different architectures

Architecture	**MLP 1 HL**	**Pruned MLP 1 HL**	**MLP 2 HL**	**GMDH**	**ANFIS**
$\bar{R}_m$	0.0285	0.0252	0.0306	**0.0238**	0.3180
σ	0.0102	0.0091	0.0142	**0.0109**	0.2862

HL-hidden layer

Table 4 shows the robustness of 1-hidden layer MLP network while the minimum number of weights after pruning was set to several different values. Both $\bar{R}_m$ and σ values are the lowest when the minimum number of weights is 400. The comparison result indicates that the minimum number of weights has an influence on the resulting robustness and should be set to an appropriate value to obtain a better performance. The best value found for the minimum number of is 400.

Table 4. Robustness of MLP (1HL) with OBS pruning

Minimun number of weights	**50**	**200**	**400**	**586 (fully connected)**
$\bar{R}_m$	0.0323	0.0288	**0.0252**	0.0285
σ	0.0279	0.0158	**0.0091**	0.0102

Table 5. Actual number of weights after OBS pruning. Note that each reported value is an average and hence it is not an integer number

Minimun number of weights	**50**	**200**	**400**
Actual number of weights	49.89	223.27	413.63

The actual number of weights after OBS pruning are listed in Table 5. It is the average number of weights for all the trials.

5. CONCLUSIONS

In this paper, a robustness measure based on a kind of leave-one-out approach has been proposed to compare the robustness and consistency of different neural network architectures used for the prediction of LOCA break size in nuclear power plants. Five different architectures with different learning algorithms were investigated to determine the most robust and efficient network for transient identification. From the results, it can be concluded that the GMDH network is the most robust architecture amongst the investigated approaches. It is also evident from the results that the OBS pruning method is able to improve the robustness of fully connected single-layer MLP neural network.

ACKNOWLEDGEMENT

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant (Grant No. EP/M018709/1).

REFERENCES

Moshkbar-Bakhshayesh, K., & Ghofrani, M. B. (2013). Transient identification in nuclear power plants: A review. Progress in Nuclear Energy, 67, 23–32. https://doi.org/10.1016/j.pnucene.2013.03.017

Lee, S. H., No, Y. G., Na, M. G., Ahn, K. I., & Park, S. Y. (2011). Diagnostics of Loss of Coolant Accidents Using SVC and GMDH Models. IEEE Transactions on Nuclear Science, 58(1), 267–276. https://doi.org/10.1109/TNS.2010.2091972

Lu, B., & Upadhyaya, B. R. (2005). Monitoring and fault diagnosis of the steam generator system of a nuclear power plant using data-driven modeling and residual space analysis. Annals of Nuclear Energy, 32(9), 897–912. https://doi.org/10.1016/j.anucene.2005.02.003

Lee, S. H., No, Y. G., Na, M. G., Ahn, K. I., & Park, S. Y. (2011). Diagnostics of Loss of Coolant Accidents Using SVC and GMDH Models. IEEE Transactions on

Nuclear Science, 58(1), 267–276. https://doi.org/10.1109/TNS.2010.2091972

da Costa, R. G., Mol, A. C. de A., de Carvalho, P. V. R., & Lapa, C. M. F. (2011). An efficient Neuro-Fuzzy approach to nuclear power plant transient identification. Annals of Nuclear Energy, 38(6), 1418–1426. https://doi.org/10.1016/j.anucene.2011.01.027

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks, 5(6), 989–993. https://doi.org/10.1109/72.329697

Hassibi, B., Stork, D. G., & Wolff, G. J. (1993). Optimal Brain Surgeon and general network pruning. In IEEE International Conference on Neural Networks (pp. 293–299 vol.1). https://doi.org/10.1109/ICNN.1993.298572

Norgaard, M., Ravn, O., Poulsen, N. K., and Hansen, L. K., (2000). Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner’s Handbook, (Springer-Verlag, London), pp. 102-111.

Jang, J. (1993). ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics, 23(3), 665-685. http://dx.doi.org/10.1109/21.256541

MathWorks, (2016). Fuzzy Logic Toolbox™: User's Guide (R2016a). Retrieved March 10, 2017 from http://cn.mathworks.com/help/pdf_doc/fuzzy/fuzzy.pdf

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics, SMC-15(1), 116–132. https://doi.org/10.1109/TSMC.1985.6313399

Celikyilmaz A., & Türksen I. B. (2009) Modelling uncertainity with fuzzy logic: with recent theory and applications. Germany: Springer.

Santosh, T. V., Srivastava, A., Sanyasi Rao, V. V. S., Ghosh, A. K., & Kushwaha, H. S. (2009). Diagnostic system for identification of accident scenarios in nuclear power plants using artificial neural networks. Reliability Engineering & System Safety, 94(3), 759–762. https://doi.org/10.1016/j.ress.2008.08.005

Santhosh, T. V., Kumar, M., Thangamani, I., Srivastava, A., Dutta, A., & Verma, V. et al. (2011). A diagnostic system for identifying accident conditions in a nuclear reactor. Nuclear Engineering and Design, 241(1), 177-184. http://dx.doi.org/10.1016/j.nucengdes.2010.10.024