A study on the robustness of neural network models for predicting the break size in LOCA

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

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

[2]Reactor Design & Development Group, Bhabha Atomic Research Centre, Mumbai, 400 085, India

Abstract

The diagnosis of loss of coolant accidents in nuclear reactors has attracted a great deal of attention in condition monitoring of nuclear power plants given that the health of the cooling system is crucial to the nuclear reactor's stable operation. Many different types of neural networks have commonly been applied to loss of coolant accident diagnosis. 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 break sizes that are not included in the training data sets. The robustness metric proposed in our previous work is applied to compare the robustness of different diagnostic models. 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 a nuclear reactor, following a simulated loss of coolant accident in a high-fidelity simulator. Given the simulation data for different break sizes, four different neural network architectures are investigated and their properties are compared and discussed. These models include a fully-connected multilayer perceptron with one hidden layer, a multilayer perceptron with one hidden layer that is pruned using the optimal brain surgeon algorithm, a fully-connected multilayer perceptron with two hidden layers, and a group method of data handling neural network. In this paper, an interpolation pre-processing method is investigated and shown to be effective to further improve the capability of neural networks for robustly predicting the break size of a loss of coolant accident. Both linear interpolation and cubic spline interpolation are studied as alternatives for the pre-processing approach. The performance of models developed with and without interpolation pre-processing are compared with the previously proposed robustness metric. Moreover, three blind cases are introduced to evaluate and compare the performance of the diagnostic models. Finally, a combined diagnostic model is proposed based on three different architectures to obtain high prediction accuracy and good robustness.

Keywords: Diagnostic models; loss of coolant accident; nuclear power plant; model robustness

1. Introduction

The reactor coolant system is the key part of nuclear power plants (NPPs). There is a critical part of the cooling system where 'breaks' can occur - these breaks are essentially ruptures of pipes leading to leaks of the coolant. The bigger the leak, the worse the problem is. These breaks, commonly denominated loss of coolant accidents (LOCA) may lead to serious consequences for the plant, the environment, and people's health and safety. To safely deal with such scenarios, every nuclear plant is equipped with an Emergency Core Cooling System (ECCS), which is highly reliable and operates automatically if it detects a break. It is immensely important for operators to detect breaks and understand their severity so they can take appropriate actions, in the unlikely case that the ECCS fails to operate.

The Three Mile Island accident revealed that operators may not efficiently handle voluminous information in abnormal conditions. Therefore, the timely and accurate recognition of NPP status requires automation development to guarantee safe and reliable operation (Mo et al., 2007). If a LOCA occurs in a nuclear power plant, it is quite difficult for operators to interpret the trends of measured variables because of the large volume of information from sensors and the fact that changes may occur rapidly when a plant moves from a normal state to an abnormal state (Moshkbar-

Bakhshayesh & Ghofrani, 2013). Therefore, many artificial intelligence techniques including neural networks have been applied to assist the operators to detect and diagnose the break size of LOCA.

To detect and diagnose the LOCA, a great deal of attention has been paid to the monitoring of the coolant system (Mo et al., 2007; Moshkbar-Bakhshayesh & Ghofrani, 2013; Choi et al., 2017). Da Costa et al. (da Costa et al., 2011) developed an operator support system based on Neuro-Fuzzy approach to identify accidents, including LOCA, rapidly and accurately. Choi et al. (Choi et al., 2017) presented a method to estimate the LOCA break size using the cascaded fuzzy neural network model. Na et al. (Na et al., 2004) applied probabilistic neural network to classify accidents such as LOCA, loss of feedwater station blackout, and steam generator tube rupture, and used a fuzzy neural network to identify these severe accident scenarios. Multilayer perceptrons (MLPs) with different network structures and learning algorithms are the most popular neural networks for fault diagnostics in NPPs (G. Vinod et al., 2003; Moshkbar-Bakhshayesh & Ghofrani, 2013). Radial basis function (RBF) networks were used by Renders et al. (Renders et al., 1995) to detect incipient incidents. Probabilistic neural networks (PNNs) and learning vector quantization (LVQ) networks were used for ‘‘Don’t Know’’ classification in (Bartal et al., 1995). Na et al. (Na et al., 2004) and da Costa et al (da Costa et al., 2011) used fuzzy neural networks (FNNs) for accident classifications. Lee et al (Lee et al., 2011) proposed a scheme to diagnose LOCAs using support vector classification (SVC) and group method of data handling (GMDH) models. The simulation results confirmed that the proposed SVC model can discover the break location without a misclassification and the proposed GMDH models can estimate the break size accurately. To detect the cracks and leakage which may appear before LOCA, Zhang et al. applied a three-layer back propagation network and a genetic neural network to predict the leak before break (LBB) for various conditions (Zhang et al., 2017). Most current diagnostic systems focus on classifications. Few of these systems were designed to predict the severity of the diagnosed scenario (Lin et al., 1995). This article focuses on the study of numerical values that measure the severity of an accident, rather than a qualitative prediction of severity.

Dynamic neural networks have been commonly applied for time series prediction. However, in this work, a static neural network with constant weights is used to map the instantaneous values of the set of input variables to the predicted break size. Normally, to train a dynamical model the training data is required to capture how the output of the system changes with the changes in the input. In our case, the available input-output data is not suitable to train dynamical models, as the training output is a constant for each break size. Therefore, dynamic neural networks are not suitable for predicting the break size in LOCA.

MLPs may have difficulties with generalization when the training data is limited (Hagan & Menhaj, 1994). It is important to select an optimal architecture for the neural network that delivers robust results. To achieve this purpose, 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 weights are eliminated one at a time according to well defined criteria, until an architecture that is optimal in some sense has been reached. An example of this approach is the use of the optimal brain surgeon (OBS) algorithm (Hassibi et al., 1993). Other strategies go in the opposite direction by starting with small network architecture and then gradually growing it, such as a GMDH approach.

In our previous work (Tian et al., 2017), we have presented a robustness measure for designing robust neural networks for LOCA break size predictions. Based on this robustness measure, this paper introduces interpolation approaches for data processing as a way to improve the robustness of the models. Moreover, blind cases are used for evaluating the developed models. The paper is organized as follows: Section 2 introduces the neural network structures that are investigated in this paper. Section 3 details the data description, modeling and testing process, along with the proposed robustness measure. Section 4 shows the data processing results and discussion. Section 5 presents the

validation of the diagnostic models and proposes a robust combined diagnostic model, followed by conclusions in Section 6.

2. Methodology

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

2.1 Multilayer perceptron

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

The output of the hidden neuron *j* can be written as

$$y_j = \Phi_j\left(\mathbf{w}_j^{\mathrm{T}}\mathbf{x} + b_j\right) = \Phi_j\left(\sum_{i=1}^{N} w_{ji}x_i + b_j\right) \tag{1}$$

Here, the input vector to the neural network is $\mathbf{x} = \left[x_1, x_2, \mathbf{L}, x_N\right]^{\mathrm{T}}$, $\mathbf{w}_j = \left[w_{j1}, w_{j2}, \mathbf{L}, w_{jN}\right]^{\mathrm{T}}$ denotes the weight vector between the hidden neuron *j* and the inputs, b_j is the bias, and Φ_j is the activation function which is normally taken as a non-linear function, such as a sigmoid function. The output of the neural network is given by Eq. (2) (Laurene, 1994; Simon, 1998).

$$\hat{y} = \Phi_o\left(\sum_{j=1}^{M} w_{oj}y_j + b_o\right) \tag{2}$$

Here, Φ_o is the activation function of the output neuron which is normally taken as a linear function, $\hat{y}$ is the final output of the MLP network, b_o is the bias value of the output neuron o, and w_{oj} is the synaptic weight value from the hidden neuron j to the output neuron o.

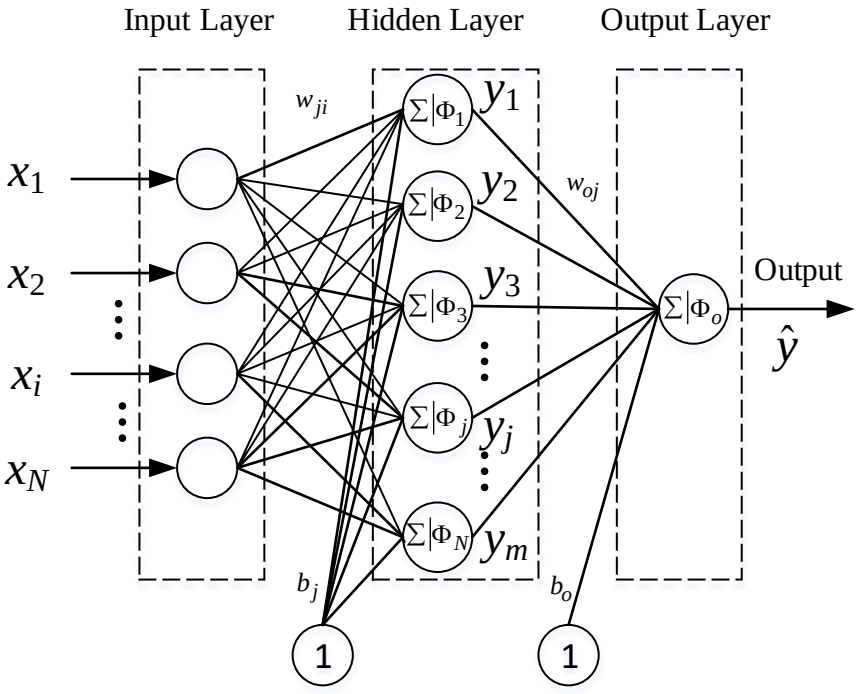

Figure 1. One hidden layer MLP with one output

A MLP is often trained with the widely used backpropagation algorithm. The algorithm consists of two steps. In the forward pass, the predicted outputs are calculated corresponding to the given inputs. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The network weights can then be adapted using any gradient-based optimisation algorithm. The whole process is iterated until the weights have converged (Laurene, 1994) or a given stopping criterion is satisfied.

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

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

Here, k is the index of iterations, $\mathbf{w}_k$ is the weight vector of the kth training iteration (which is defined in Eq. (4)), $\mathbf{I}$ is the identity matrix, μ is the step size, e_k is the error value of the kth training iteration between the output and target.

$$\mathbf{w}_k = \left[w_{k,1}, w_{k,2}, \mathbf{L}\ , w_{k,Z} \right]^{\mathrm{T}} \tag{4}$$

where, Z is the total number of weights.

$$\mathbf{e}_k = \left[e_{k,1}, e_{k,2}, \mathbf{L}\ , e_{k,P} \right]^{\mathrm{T}} \tag{5}$$

where, P is the number of input patterns.

$\mathbf{J}_k$ is a Jacobian matrix of the kth training iteration (which can be seen defined in Eq. (6)).

$$\mathbf{J}_k = \begin{bmatrix} \frac{\partial e_{k,1}}{\partial w_{k,1}} & \frac{\partial e_{k,1}}{\partial w_{k,2}} & \mathbf{L} & \frac{\partial e_{k,1}}{\partial w_{k,Z}} \\ \frac{\partial e_{k,2}}{\partial w_{k,1}} & \frac{\partial e_{k,2}}{\partial w_{k,2}} & \mathbf{L} & \frac{\partial e_{k,2}}{\partial w_{k,Z}} \\ \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} \\ \frac{\partial e_{k,P}}{\partial w_{k,1}} & \frac{\partial e_{k,P}}{\partial w_{k,2}} & \mathbf{L} & \frac{\partial e_{k,P}}{\partial w_{k,Z}} \end{bmatrix} \tag{6}$$

2.2 Optimal brain surgeon algorithm

In neural networks, the regularization problem is often cast as minimizing the number of connection weights. When there are too many weights, overfitting problems can occur and this results in poor generalization performance. Conversely, if there are too few weights, the network might not capable to adequately learn the key aspects of the training data. The optimal architecture should be large enough to learn the problem and small enough to generalize well. When the training set is very limited, it is especially important to choose the network architecture wisely in that it should contain only the most essential weights. The architecture selection is in this case can be difficult since the limitations in the data set cause difficulties to set aside a suitable data set for testing purposes. The OBS algorithm proposed by Hassibi et al. (Hassibi et al., 1993) is an effective method to optimize the network architecture, allowing to improve the generalization ability of the network and hence its robustness.

The OBS algorithm judges the importance of a weight by its saliency which represents the increase in error that introduced by eliminating it. Then, the OBS algorithm deletes the weights with the weakest saliency, which means they are insignificant. This is an iterative process which continues until a

stopping criterion is satisfied. The procedure of OBS algorithm is summarized in **Error! Reference source not found.**.

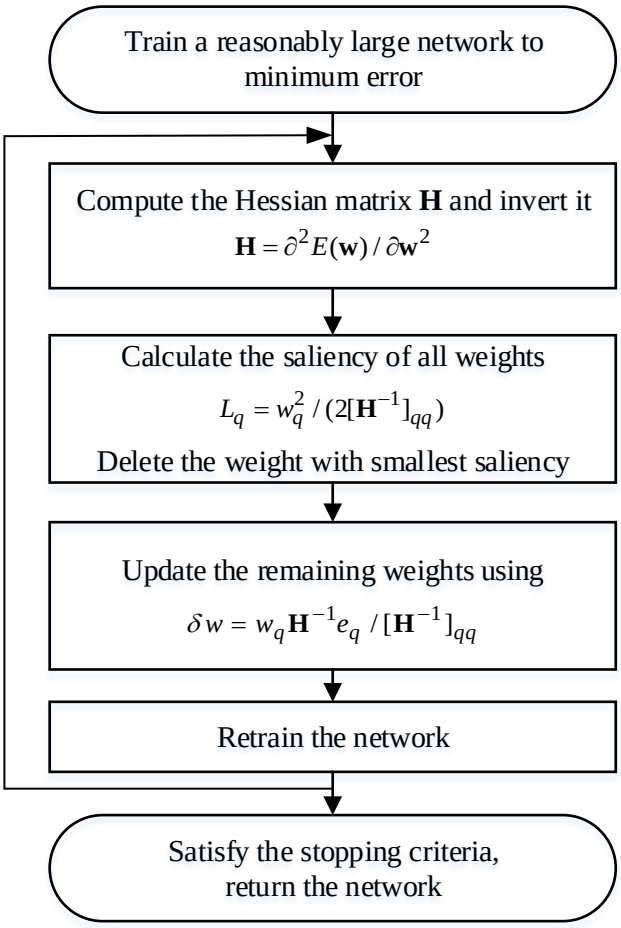

Figure 2. The process flow chart of OBS

The saliency of weights q can be calculated by Eq. (7).

$$L_q = \frac{w_q^2}{2[\mathbf{H}^{-1}]_{qq}} \tag{7}$$

Here, w_q denotes the value of weights q, $\mathbf{H}^{-1}$ is the inverse of the Hessian matrix $\mathbf{H}$, which is the second order derivative of the error function with respect to all weights as follows

$$\mathbf{H} = \begin{bmatrix} \frac{\partial E^2}{\partial w_1 \partial w_1} & \frac{\partial E^2}{\partial w_1 \partial w_2} & \mathbf{L} & \frac{\partial E^2}{\partial w_1 \partial w_Z} \\ \frac{\partial E^2}{\partial w_2 \partial w_1} & \frac{\partial E^2}{\partial w_2 \partial w_2} & \mathbf{L} & \frac{\partial E^2}{\partial w_2 \partial w_Z} \\ \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} \\ \frac{\partial E^2}{\partial w_Z \partial w_1} & \frac{\partial E^2}{\partial w_Z \partial w_2} & \mathbf{L} & \frac{\partial E^2}{\partial w_Z \partial w_Z} \end{bmatrix} \tag{8}$$

After eliminating the weight with weakest saliency, the remaining weights are updated according to:

$$w_q \leftarrow w_q + \frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} e_q \tag{9}$$

After updating the weights, the pruned network is retrained to avoid any loss of performance due to pruning. If the stopping criterion is not satisfied, the weight pruning process is repeated. Further details about the optimal brain surgeon algorithm and its derivation are given in reference (Hassibi et al., 1993).

2.3 Group method of data handling network (GMDH)

GMDH (Lu & Upadhyaya, 2005) is an approach that involves growing a neural network which has a self-organized structure. There is no need to specify the number of layers and neurons in each layer. The GMDH algorithm replaces the neuron activation functions by the set of hierarchically connected partial models. Coefficients of these models are estimated by the least squares method. GMDH networks gradually increase the number of partial model components and find a model structure with optimal complexity. Thus this approach avoids the tedious work of architecture selection, and allows focusing on the organization of effective model inputs based on physical and statistical considerations. Figure 3 shows the structure of a GMDH network with four inputs and one output.

The algorithm is based on a multilayer structure using the general form, which is referred to as the Kolmogorov-Gabor polynomial (Volterra functional series) (Onwubolu, 2015)

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k \mathbf{K} \tag{10}$$

where the external input vector is represented by $X = [x_1, x_2, \mathbf{K}]$, y is the corresponding output value, and a_i, a_{ij}, and a_{ijk} are coefficients. The polynomial equation represents a full mathematical description. The whole system of equations can be represented using a matrix form as shown below:

$$y = f(X) \tag{11}$$

where

$$X = \begin{bmatrix} x_{11} & x_{12} & \mathbf{L} & x_{1m} \\ x_{21} & x_{22} & \mathbf{L} & x_{2m} \\ \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} \\ x_{n1} & x_{n2} & \mathbf{L} & x_{nm} \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \mathbf{M} \\ y_n \end{bmatrix} \tag{12}$$

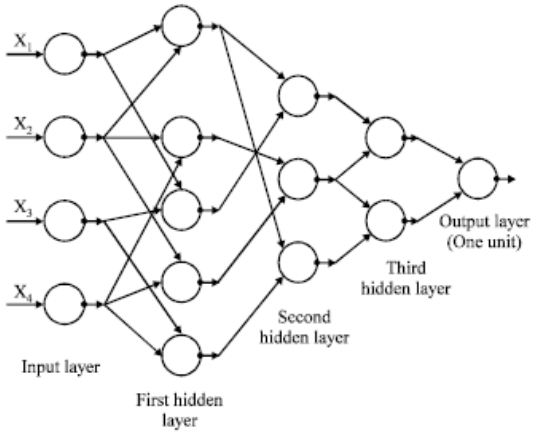

Figure 3. Structure of a GMDH network

The output of a neuron with three inputs shown in Figure 4 can be expressed by a partial polynomial, for example as given in equation (13).

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1^2 + a_8x_2^2 + a_9x_3^2 + a_{10}x_1x_2x_3 + a_{11}x_1^3 + a_{12}x_2^3 + a_{13}x_3^3 + \ldots \quad (13)$$

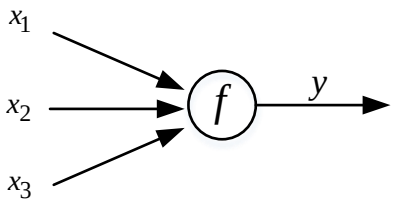

Figure 4. A neuron of a GMDH network with three inputs

3. Data description and processing

3.1 Database generation

The simulated data for LOCA used in this study is described in (Santosh et al., 2009; Santosh et al., 2011). The database generation needed the integration of various computational tools which are described in (Santhosh et al., 2011). The thermal-hydraulic response of the reactor core and the primary heat transport with LOCA were generated using the RELAP5 thermal hydraulic code (Fletcher & Schultz, 1995; Santosh et al., 2009). 37 measurable variables related to LOCAs have been used for diagnostic modeling and testing. Those variables have been identified to be suitable input parameters after conducting a detailed study on the event progression of LOCA scenario, which include the fast sensing pressure signals, differential pressure signals and other critical parameters related to containment thermal hydraulics. Full details on the parameters, range of values, etc. has been described in the reference (Santosh et al., 2009). The prediction of LOCA break size is modeled as a regression problem where a set of input values are 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. To detect the LOCA as quickly as possible in the earlier phase of its development, the time scale of the transient duration is divided into three intervals to enable the minimization of the negative consequences of the break. At the same time, the time scale is chosen so 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 a large break LOCA event (Santosh et al., 2009 & Santosh et al., 2011). The time scale for a transient period of 60s is as follows:

$$\Delta t = \begin{cases} 0.01 & for \quad t \quad 0s \quad to \quad 2s \\ 0.1 & for \quad t \quad 2.1s \quad to \quad 30s \\ 0.5 & for \quad t \quad 30.5s \quad to \quad 60s \end{cases} \quad (14)$$

where t is the time and Δt is the time increment.

Break sizes ranging from 20% to 200% (including 20%, 60%, 100%, 120% and 200%) have been modeled using four different network architectures. The break area is represented in terms of the percentage of the maximum possible break area from a double-ended guillotine break of the reactor header. The data is obtained when the LOCA event occurs in the reactor inlet header (RIH) with the availability of the ECCS.

For illustration, Figure 5 shows the time series of the pressure in north inlet header under six different conditions. It is apparent that the inlet header pressure first increases and then decreases rapidly when the LOCA occurs. The larger the break size, the faster the change speed and the lower the pressure value. The amplitude of pressure exhibits fluctuation while decreasing. The pressure values of break

sizes between 60% and 200% are quite similar during the first few seconds of LOCAs. Due to the fluctuation and the high similarity for the first few seconds, it is difficult to distinguish the different break sizes by amplitude especially at the early stage of LOCAs.

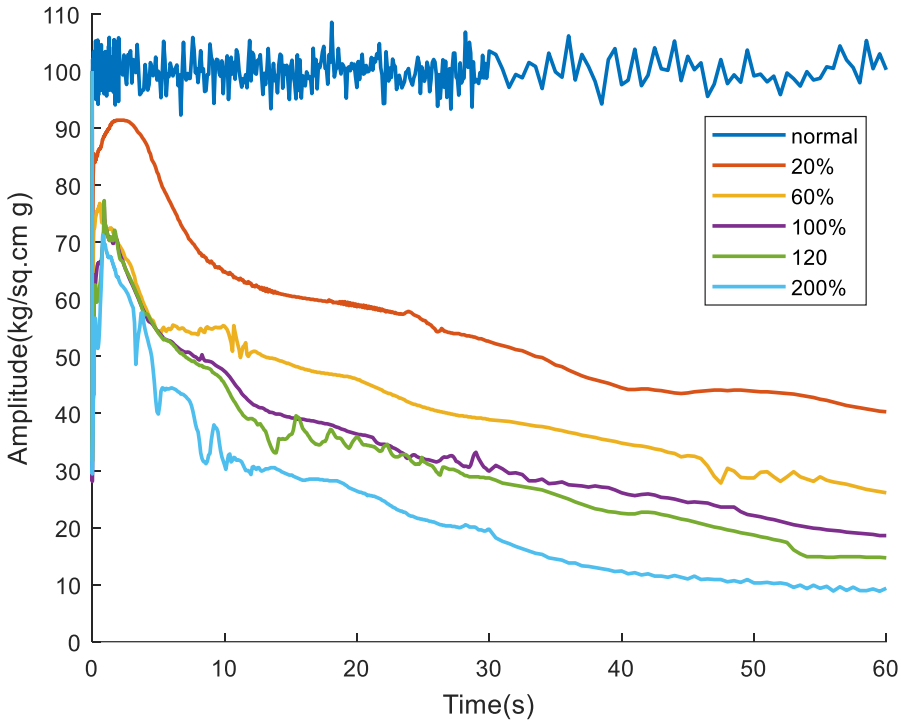

Figure 5. The pressure in north inlet header for different break sizes (normal, 20%, 60%, 100%, 120%, and 200%)

3.2 Modelling and testing process

The modelling and testing process is presented in Figure 6. The procedure runs *N* 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. A value of *N*=50 is used in this work as it provides an acceptable balance between computational load and accuracy of the mean values. 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 during training process).

3.2.1 Leave-one-out method

In machine learning, the leave-one-out cross-validation method (Reich & Barai, 1999) uses a single observation from the original sample as the test data, and the remaining observations as the training data. This is repeated so that each observation in the sample is used once as the test data.

Inspired by the classical leave-one-out cross validation method, in this work we have used a testing method that we called leave-one-out training/testing, where we leave out from the training and validation sets all the 37 time-series corresponding to one particular break size, but the testing set includes the data corresponding to the break size which was left out, as well as unseen samples from all other break sizes. In other words, we only leave out one break size in the training and validation sets while the testing data includes data corresponding to all break sizes. All the data sets for training are listed in Table 1. Test 1 is applied to train the normal models for blind case validation and Test 2 to Test 5 are applied to train the models that are used for robustness comparison.

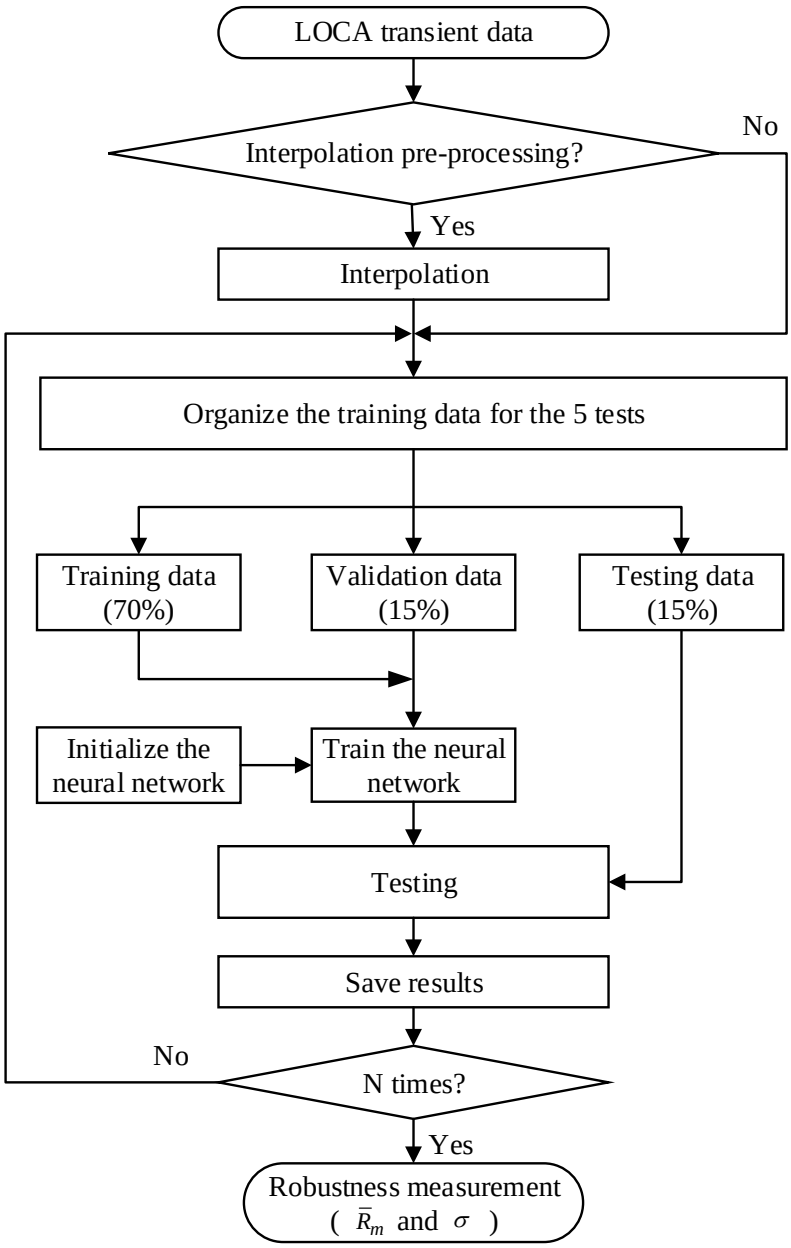

Figure 6. Data processing flowchart

Table 1. List of training data sets

Test No.	Training sets
1	All available break sizes
2	Leave out 20% break
3	Leave out 60% break
4	Leave out 100% break
5	Leave out 120% break

3.2.2 Interpolation pre-processing

The training data sets are only available for a limited number of break sizes as shown in Table 1. This causes robustness issues, as the resulting neural network models are not very accurate for predicting unseen break sizes. With the aim of improving robustness, we propose to use interpolation to generate data sets for additional break sizes for neural network training. The interpolated data is not as accurate as the data generated from a high-fidelity simulator, but it helps improving the robustness of neural network models. For each interpolated break size, the time-series for every measured variable is interpolated between two adjacent break sizes.

For the leave one out tests, the interpolation should be based on the remaining data sets after leaving out the data set for a particular break size. The remaining break sizes and interpolated break sizes for all tests are listed in Table 2.

Table 2. The interpolated break sizes

Test	Left out break size (%)	Remaining break sizes (%)	Interpolated break sizes (%)
1		0, 20, 60, 100, 120, 200	10, 40, 80, 110, 160
2	20	0, 60, 100, 120, 200	15, 30, 45, 80, 110, 160
3	60	0, 20, 100, 120, 200	10, 40, 60, 80, 110, 160
4	100	0, 20, 60, 120, 200	10, 40, 75, 90, 105, 160
5	120	0, 20, 60, 100, 200	10, 40, 80, 125, 150, 175

To make the results comparable, the total number of remaining break sizes and interpolated break sizes is the same for all the tests, so the number of break sizes included in the training data of each test is the same.

Overall, our criteria for selecting the interpolated sizes is that a new generated break size can split the interval between two adjacent break sizes equally. If the break sizes are $B_s(1)$, $B_s(2)$,.., $B_s(n)$, the interpolated break size can be calculated as

$$B_{sn}(i) = [B_s(i) + B_s(i+1)]/2, \quad i = 1,\mathbf{K}\, n-1 \tag{15}$$

Therefore, the new break size list is $B_s(1), B_{sn}(1), B_s(2), B_{sn}(2),\mathbf{K}\, B_s(n-1), B_{sn}(n-1), B_s(n)$.

For the leave one out tests, the selection of interpolated break sizes is the same with Test No. 1 for the intervals unrelated to the left-out break size. Taking the case leaving out 20% break size as an example (see Test No. 2 in Table 2), the interpolated break sizes 80%, 110%, and 160% represent the average of the two adjacent break sizes within the range from 60% to 200%. In order to keep the total number of break sizes uniform, three break sizes are interpolated within the range [0 60]. These three break sizes (15%, 30%, and 45% in Test 2) are selected to split the range [0 60] into four equal parts.

Two well-known interpolation methods were applied in this study: linear interpolation and cubic spline interpolation.

3.2.2.1 Linear interpolation

Assume we are given a set of data points $(x_1, y_1),\mathbf{K}\,(x_n, y_n)$ where $\alpha = x_1 < \mathbf{L} < x_n = \beta$ and $y_i = f(x_i), i = 1:n$. The piecewise linear interpolant is built upon the local linear interpolants as

$$L_i(x) = a_i + b_i(x - x_i) \tag{16}$$

where for $i = 1:n-1$ the coefficients are defined by $a_i = y_i$ and $b_i = \dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$.

Note that $L_i(x)$ is just the linear interpolant of f within the interval $[x_i, x_{i+1}]$. Then define

$$L(x) = \begin{cases} L_1(x) & if \quad x_1 \le x < x_2 \\ L_2(x) & if \quad x_2 \le x < x_3 \\ \mathbf{M} & \mathbf{M} \\ L_{n-1}(x) & if \quad x_{n-1} \le x \le x_n \end{cases} \tag{17}$$

Figure 7 illustrates an example of linear interpolation. As shown in the graphs, one break size is interpolated between two adjacent sizes. Figure 7(b) is an enlarged detailed graph of the rectangular area in Figure 7(a) and it can be clearly seen that the interpolated points are in the middle of the corresponding points in two adjacent sizes. With the interpolated datasets, there are more break sizes that can be used for neural network training. Our experience is that neural networks have good performance in predicting break sizes that are included in the training data set, but performance is less good for unseen break sizes that are not close to break sizes in the training data set. Our hypothesis is that by reducing the gaps between break sizes in the training data sets, the proposed interpolation technique will help to improve the robustness and generalization performance of the neural network.

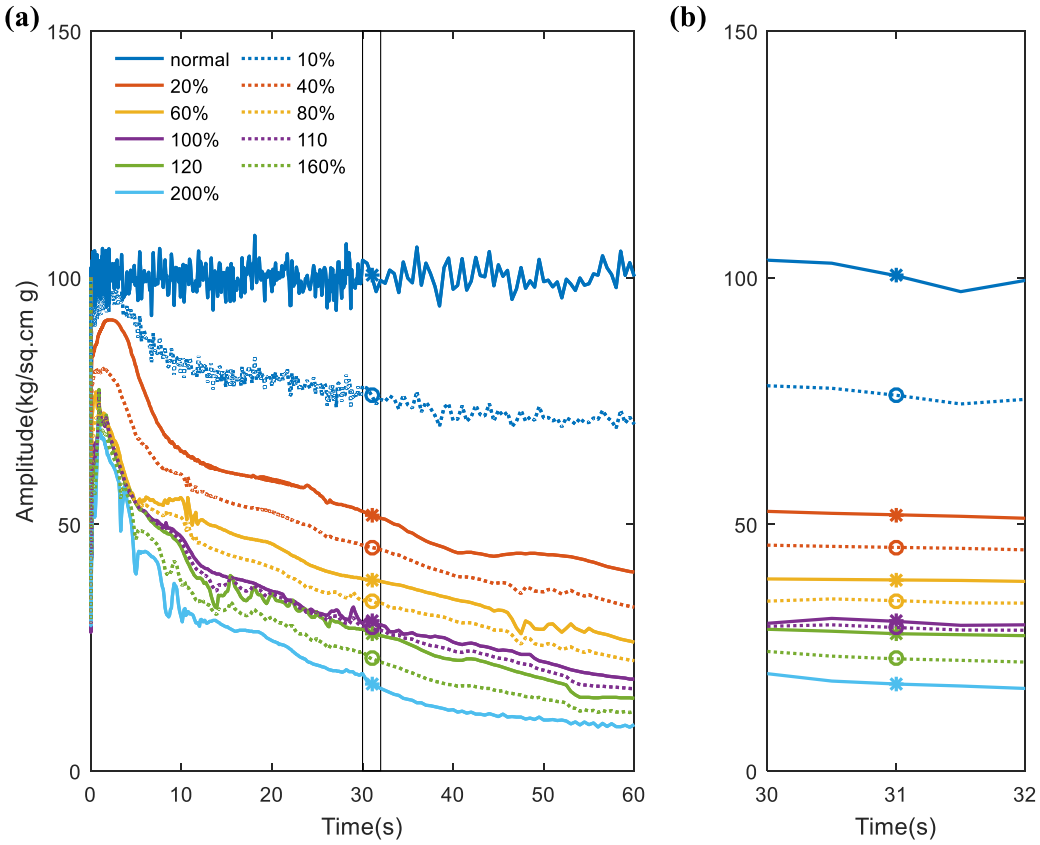

Figure 7. Illustration of the transient data
Continuous cures represent original data.
Dotted curves represent interpolated data by linear interpolation

3.2.2.2 Cubic spline interpolation

Cubic splines are piecewise cubic functions with continuous first and second derivatives. Cubic spline approximants offer a reasonable balance between the smoothness of polynomial approximants and the flexibility of linear spline approximants, and typically produce good approximations for both the function and its first and second derivatives. The interpolated value at a query point is based on a

cubic interpolation of the values at neighbouring grid points in each respective dimension. (Loan, 1999; Davis, 2014; Wang, 2018)

If we are given the n data points $(x_1, y_1), \mathbf{K}, (x_n, y_n)$ in an increasing order, where x_i is distinct, the cubic spline $S(x)$ through the data points $(x_1, y_1), \mathbf{K}, (x_n, y_n)$ will be a set of cubic polynomials:

$$S(x) = \begin{cases} S_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & if \quad x_1 \le x < x_2 \\ S_2(x) = y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & if \quad x_2 \le x < x_3 \\ S_{n-1}(x) = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_1(x - x_{n-1})^2 + d_1(x - x_{n-1})^3 & if \quad x_{n-1} \le x < x_n \end{cases} \tag{18}$$

With the following conditions (known as properties):

a. $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ for $i = 1, \mathbf{K}, n-1$
   This property guarantees that the spline $S(x)$ interpolates the given data points.

b. $S'_{i-1}(x_i) = S'_i(x_i)$ for $i = 2, \mathbf{K}, n-1$
   $S'(x)$ is continuous on the interval $[x_1, x_n]$; this property forces the slopes of neighboring parts to agree when they meet.

c. $S''_{i-1}(x_i) = S''_i(x_i)$ for $i = 2, \mathbf{K}, n-1$
   $S''(x)$ is continuous on the interval $[x_1, x_n]$, which also forces the neighboring spline to have the same curvature, to guarantee the smoothness.

d. Two end conditions (not-a-knot in Matlab):
   $S'''_1(x_2) = S'''_2(x_2)$, $S'''_{n-2}(x_{n-1}) = S'''_{n-1}(x_{n-1})$.

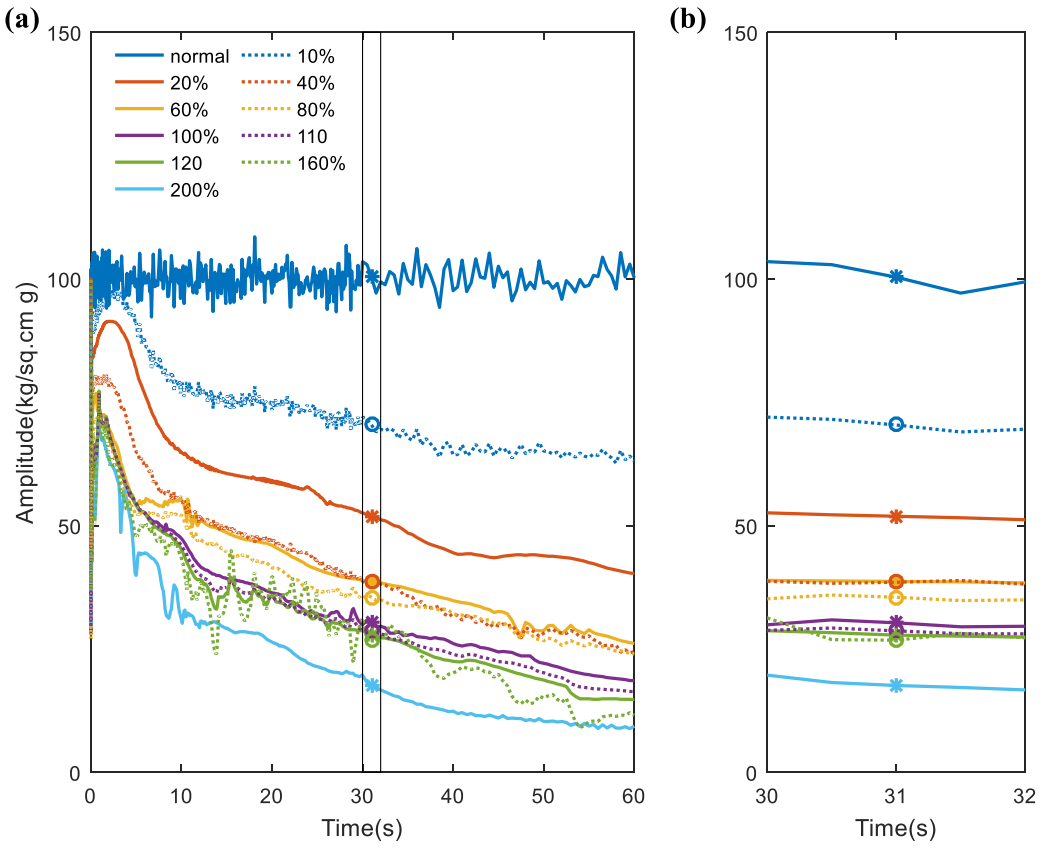

Figure 8. Illustration of the transient data (Continuous cures represent original data; Dotted curves represent interpolated data by linear interpolation)

Figure 8 illustrates an example of cubic spline interpolation. As shown in the figure, one break size is interpolated between two adjacent sizes. Figure 8(b) shows an enlarged graph of the rectangular area in Figure 8(a). Some of the interpolated points are not in the middle of the corresponding points in two adjacent sizes, which is different with linear interpolation. Given that the high-fidelity model employed to generate the LOCA signals is nonlinear, we have used the cubic spline interpolation to provide a comparison against the linear interpolation technique.

3.2.3 Parameters for initial networks

For the 1-hidden layer MLP, there are 15 neurons in the hidden layer. For the 2-hidden layer MLP, there are 19 neurons in the first hidden layer and 26 neurons in the second hidden layer. The method used to optimise the architecture of the 2-hidden layer MLP is described in (Su et al., 2003, Santhosh et al., 2007, Santhosh et al., 2018, & Tian et al., 2017). In the GMDH modelling, the structure is optimised by the method. There are 586 weights in the fully-connected 1-hidden layer MLP. The minimum number of weights is set to 400, so there are 186 weights pruned at most to guarantee the accuracy of modelling (Tian et al., 2017).
For the GMDH method, the maximum number of inputs for each neuron is 3, the degree of polynomials is 3, and the maximum number of neurons in a layer is 7 (Tian et al., 2017).

3.3 A robustness measure

The prediction results from the different models studied are presented in terms of mean square error (MSE) of the prediction. The MSE is defined as

$$E_n = \frac{1}{M}\sum_{k=1}^{K} e_{n,k}^2 \tag{19}$$

where e is the error between the neural network output and target, n and k are the index of test and sample, respectively, and K is the total number of samples. Smaller MSE values signify a better performance.

The following robustness measure R_m is proposed:

$$R_m = \max_{n\in[2,5]} E_n \tag{20}$$

where, E_n is the MSE for each leave-one-out case calculated by Equation (19). 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, each network architecture was trained and tested M=50 times independently. To compare the performance of different network architectures, the mean value of the robustness measure R_m for the M trials is calculated as:

$$\bar{R}_m = \frac{1}{M}\sum_{k=1}^{M} R_m(k), M = 50 \tag{21}$$

and the standard deviation of the robustness measure is:

$$\sigma = \sqrt{\frac{1}{M-1}\left|\sum_{k=1}^{M} R_m(k) - \bar{R}_m\right|^2} \tag{22}$$

A lower value of $\bar{R}_m$ indicates a better prediction accuracy; while a lower value of σ indicates that the prediction performance is more consistent.

4. Simulation study of the investigated architectures

4.1 Comparing the testing results with and without interpolation pre-processing

Figure 9-Figure 11 show the results for the 2-hidden layer MLP architecture where the training has been performed with three different kinds of data: original data, original data plus linear interpolation data, and original data plus cubic interpolation data. The blue lines in the graphs indicate the targets and the red dotted lines denote the outputs of neural networks. Only the test results are presented and used for calculating the robustness measure.

Figure 9(a) shows the results from the network trained with data of all break sizes. From five leave-one-out tests in Figure 9 (b)-(e), we can observe that the differences between outputs and targets for the break sizes left out during training are larger than that between outputs and targets when a particular break size is present in the training set by a factor of between 1.5 and 17. The same behavior can be observed for the results shown in Figure 10 and Figure 11.

Comparing the results in Figure 9-Figure 11, we can observe that the MSE values of the leave-one-out tests for the networks trained with interpolation pre-processing (Figure 10 and Figure 11) are smaller than the corresponding networks trained without interpolation pre-processing (Figure 9). The worst-case performance measure values R_m are also much smaller with interpolation pre-processing than without it by a factor of around 0.3. From the results in Figure 9-Figure 11, it can be concluded that the interpolation pre-processing is effective in improving the LOCA break size prediction performance of neural networks.

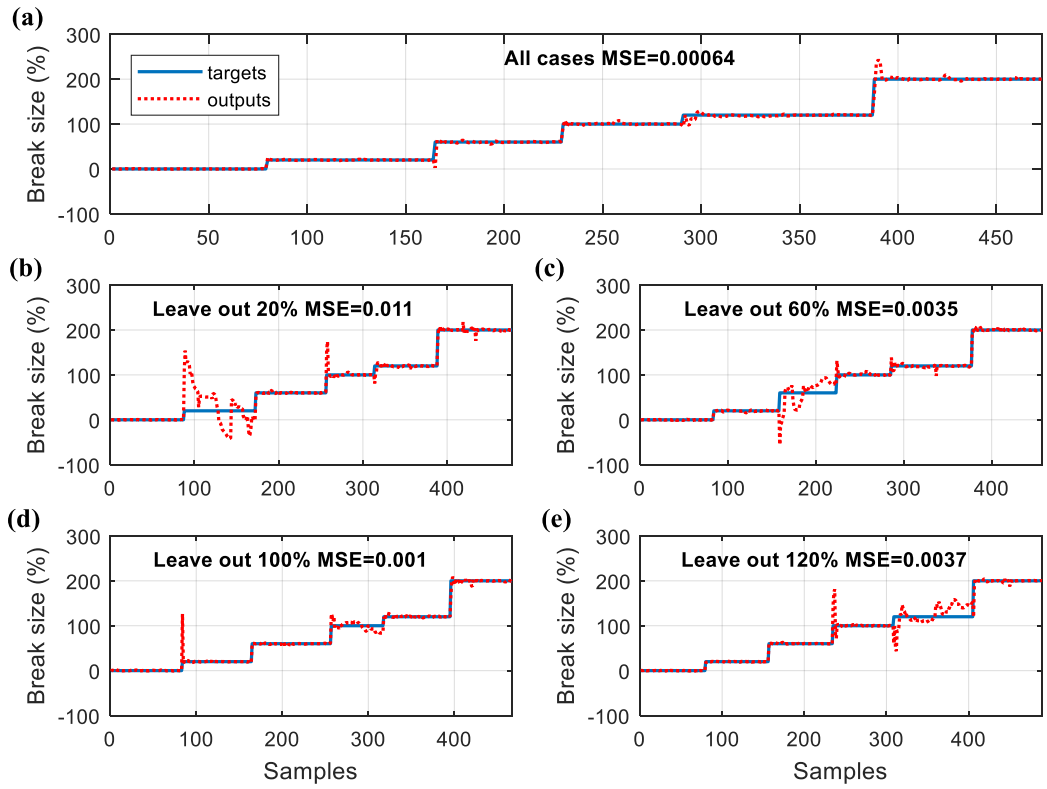

Figure 9. Results from the 2-hidden layer MLP (without pre-processing, $R_m = 0.011$)

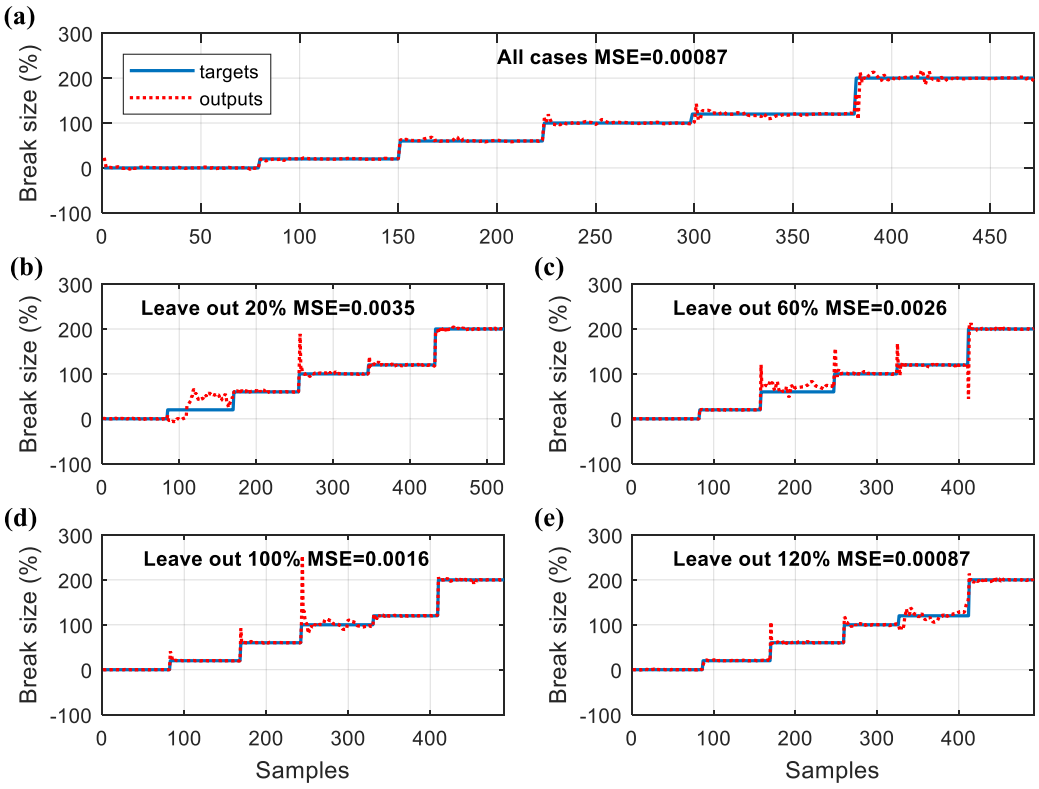

Figure 10. Results from the 2-hidden layer MLP (with linear interpolation pre-processing, $R_m = 0.0035$)

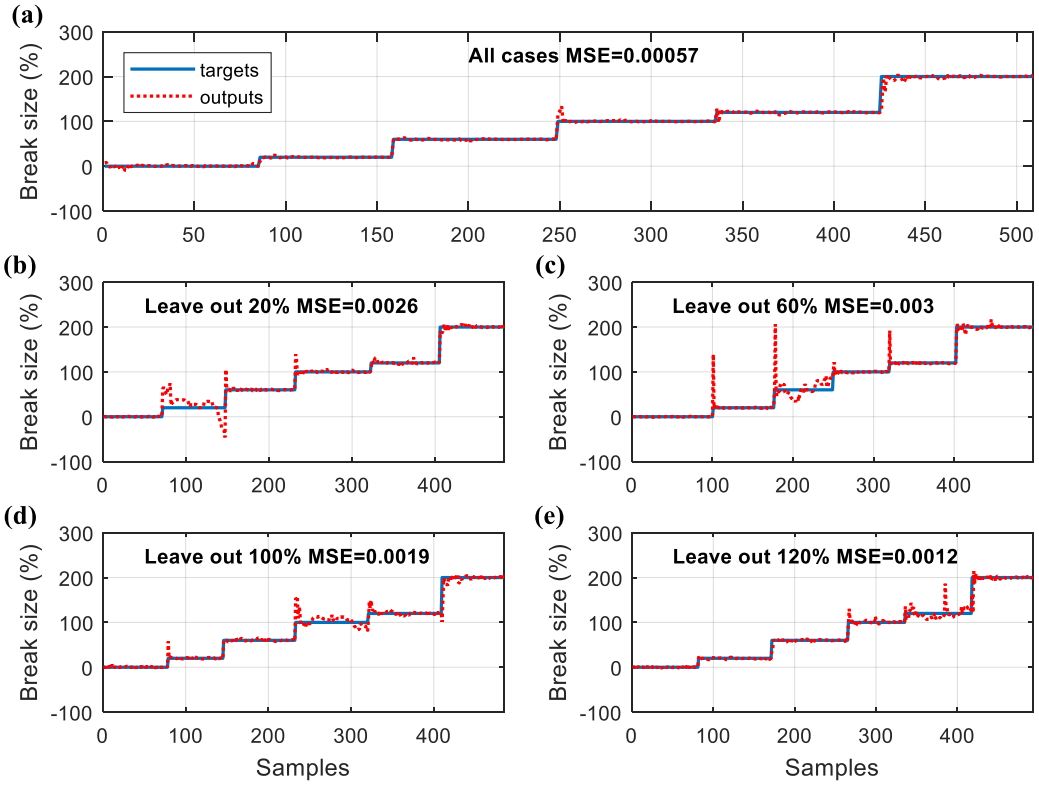

Figure 11. Results from the 2-hidden layer MLP (with cubic spline interpolation pre-processing, $R_m = 0.0030$)

4.2 Comparing the testing results before and after OBS pruning

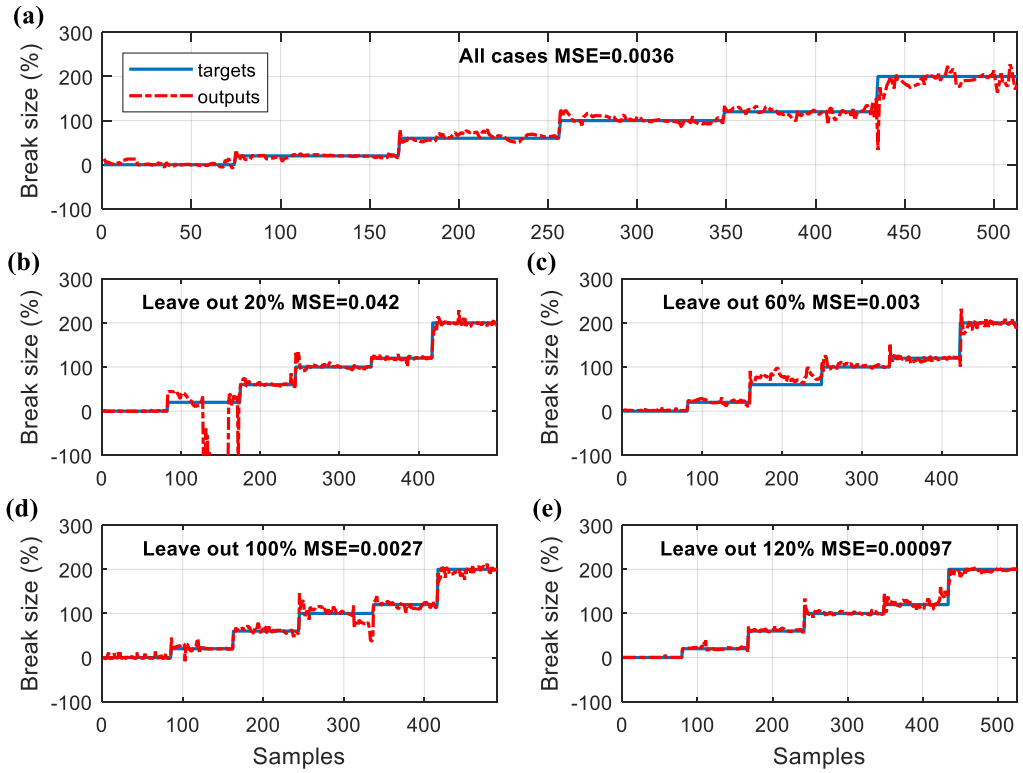

Figure 12. Results from the fully-connected 1-hidden layer MLP (with linear interpolation pre-processing, $R_m = 0.042$)

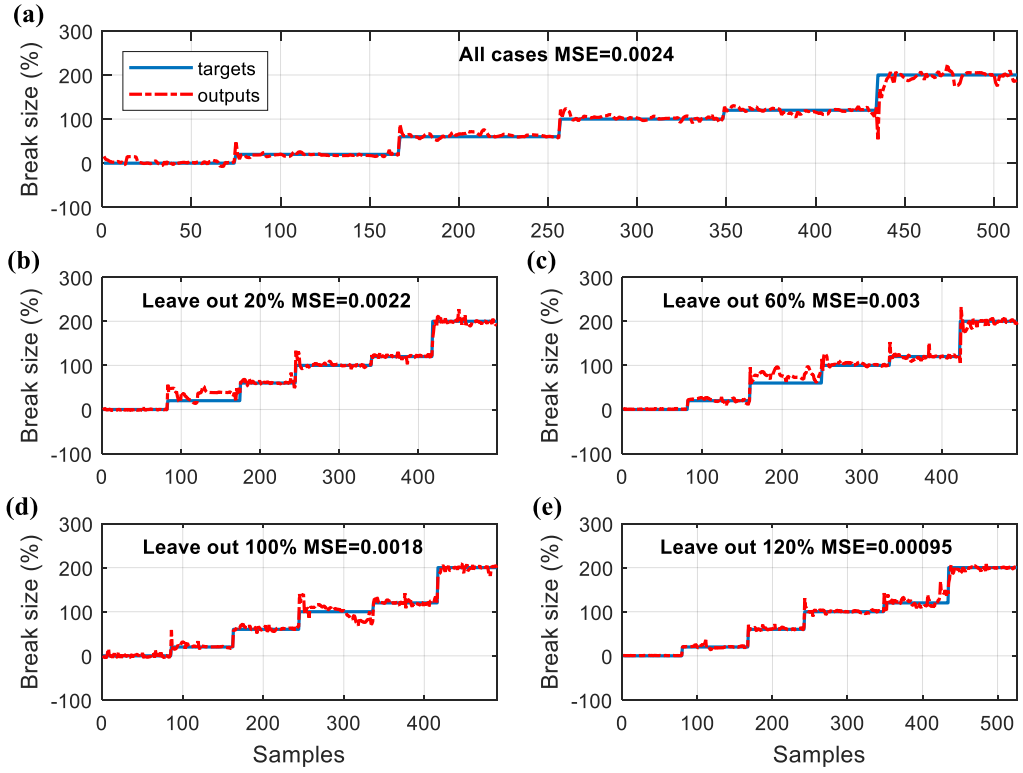

Figure 13. Results from the pruned 1-hidden layer MLP (with linear interpolation pre-processing, $R_m = 0.003$)

In this subsection, we compare the performance of the fully-connected 1-hidden layer MLP and the OBS pruned 1-hidden layer MLP. The results are obtained using linear interpolation pre-processing because this approach provided the best performance as discussed in the previous subsection.

Figure 12 and Figure 13 show the results for the fully-connected 1-hidden layer MLP and the pruned 1-hidden layer MLP networks. Comparing the outputs in the two figures, we can observe that there is less fluctuation with the pruned networks than with the corresponding fully-connected neural networks. Particularly, the predictions when leaving out 20% break size shows an improved performance after pruning. The worst value of the robustness measure R_m for the pruned network is significantly smaller by a factor of 0.07 than the corresponding value obtained with the fully-connected network. This is because the OBS pruning can improve the generalization performance of neural networks. In conclusion, it can be said that the OBS pruning is useful for improving the robustness of break size predictions.

4.3 Testing results from GMDH

Figure 14 shows the results from the GMDH networks trained with linear interpolation pre-processing. There is more fluctuation in the outputs compared to the MLP networks. However, the MSE values are larger than those obtained with the 2-hidden layer MLP network trained with linear interpolation pre-processing by a factor of between 2.4 and 6.9.

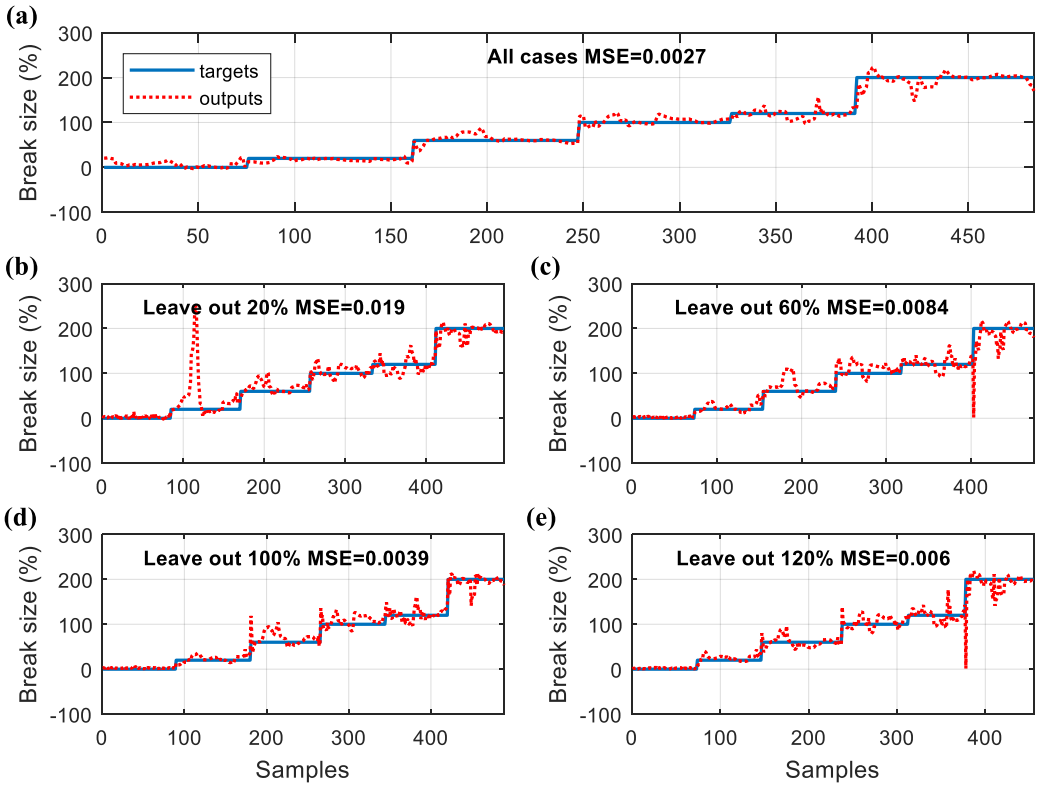

Figure 14. Results from the GMDH networks (with linear interpolation pre-processing, $R_m = 0.019$)

4.4 Summary

In this subsection, we compare the overall performances of the different architectures and pre-processing methods for predicting the break size in LOCA. Each network architecture was trained N=50 times and the robustness measure $\bar{R}_m$ was calculated according to the method described in subsection 3.3.

Table 3. Results from the networks without interpolation pre-processing

Robustness measure	**Architecture**			
	Fully-connected MLP 1 HL	Pruned MLP 1 HL	MLP 2 HL	GMDH
$\bar{R}_m$	0.0180	0.0147	0.0331	0.0236
σ	0.0115	0.0100	0.0153	0.0111

HL-hidden layer

The robustness comparison results from four 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 yielded the best robustness. The σ value for the GMDH network is slightly larger than the fully-connected 1-hidden layer MLP and the pruned 1-hidden layer MLP. Both $\bar{R}_m$ and σ values of the pruned 1-hidden layer MLP are smaller (by a factor of around 0.85) than the corresponding fully-connected network, which indicates that the OBS pruning algorithm was able to improve the robustness and consistency of the neural network.

Table 4. Results from the networks with linear interpolation pre-processing

Robustness measure	**Architecture**			
	Fully-connected MLP 1 HL	Pruned MLP 1 HL	MLP 2 HL	GMDH
$\bar{R}_m$	0.0095	0.0081	0.0057	0.0171
σ	0.0100	0.0098	0.0042	0.0046

Table 4 presents the test results from networks trained with the linear interpolation pre-processing method. Compared to the results in Table 3 (training with original data only), both $\bar{R}_m$ and σ values for all the networks trained with interpolation pre-processing are much smaller (by a factor of between 0.17 and 0.55), which indicates that the robustness and consistency have been significantly improved. Therefore, the linear interpolation pre-processing can improve both the robustness and consistency of the models. The $\bar{R}_m$ value of the pruned 1-hidden layer MLP is slightly smaller (by a factor of 0.85) than the corresponding fully-connected network. The 2-hidden layer MLP has a slightly better performance than the pruned 1-hidden layer MLP, which is different from the results from neural networks without interpolation pre-processing. In this case, GMDH has the largest $\bar{R}_m$ value, so that the performance improvement brought by the linear interpolation pre-processing for GMDH networks was not as good as for the other three types of networks.

Table 5 shows the test results from networks trained with the cubic spline interpolation pre-processing method. Both the $\bar{R}_m$ and σ values are smaller compared to the results from networks trained with the original data only, and smaller than the linear interpolation results (Table 4) as well. Therefore, it can be concluded that the cubic interpolation pre-processing can also improve the prediction accuracy and consistency and better than the linear interpolation pre-processing approach. The 2-hidden layer

MLP has better performance than the pruned 1-hidden layer MLP, which is similar to the results with linear interpolation pre-processing. GMDH still has the largest $\bar{R}_m$ value.

Table 5. Results from the networks with cubic spline interpolation pre-processing

Robustness measure	**Architecture**			
	Fully-connected MLP 1 HL	Pruned MLP 1 HL	MLP 2 HL	GMDH
$\bar{R}_m$	0.0073	0.0071	0.0049	0.0084
σ	0.0047	0.0062	0.0034	0.0011

From the results in Table 3-Table 5, it can be concluded that both linear and cubic spline interpolation pre-processing approaches can improve the robustness and consistency of models for break size predictions. The performance improvements achieved with the cubic spline interpolation pre-processing method were clearly better than those obtained with the linear interpolation pre-processing method. Amongst all the models trained with cubic spline pre-processing approach, the 2-hidden layer MLP model has the smallest $\bar{R}_m$ value while the GMDH model has the smallest σ value. This means that the 2-hidden layer MLP model trained with cubic spline pre-processing has the highest robustness and the GMDH model has the best consistency.

5. Validation of the diagnostic models with blind cases

5.1 Validation of the models with blind cases

Table 6. Description of the models

Model No.	**Architecture**	**Pre-processing**
1	Fully-connected 1-hidden layer MLP	No interpolation
2		Linear interpolation
3		Cubic spline interpolation
4	Pruned 1-hidden layer MLP	No interpolation
5		Linear interpolation
6		Cubic spline interpolation
7	2-hidden layer MLP	No interpolation
8		Linear interpolation
9		Cubic spline interpolation
10	GMDH	No interpolation
11		Linear interpolation
12		Cubic spline interpolation

To compare the prediction performance of the developed models, validation studies were carried out with three blind cases, which are transients corresponding to particular unseen break sizes. Blind case 1 has a break size value of 50%, case 2 has a break size value of 75%, and case 3 has a break size value of 160%. Each of these three cases was applied as an input to the trained networks, and the

resulting outputs were obtained for performance analysis. The trained networks had the following four architectures: 2-hidden layer MLP, fully-connected 1-hidden layer MLP, pruned 1-hidden layer MLP, and GMDH network. Three types of pre-processing were applied separately for each architecture, so there are twelve types of models in total, with the details of these models listed in

Table 6. Each model type was trained 50 times separately, as described in Section 3.2 (Test 1), with all available break sizes (20%, 60%, 100%, 120%, 200%), and the best performing model was selected for each of them according to the value of the MSE and based on the unseen test data. For the purposes of the analysis discussed in this section, the outputs from the best models found for each architecture and pre-processing type are shown separately in Figure 17 to Figure 18. Finally, the best pre-processing type is selected for each architecture and the final results are selected for comparison. To show the results clearly, the outputs of the best performing models are plotted using red color.

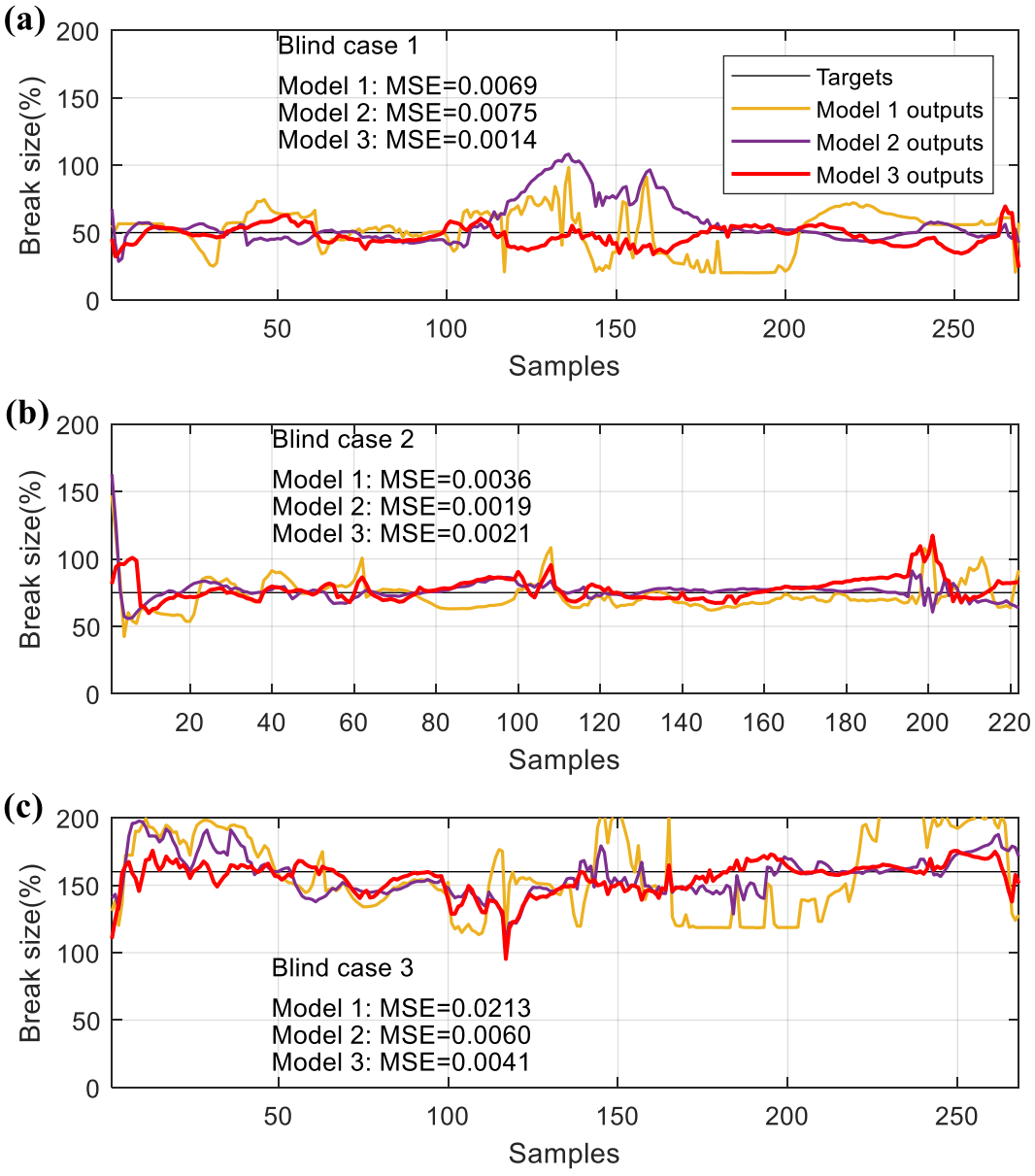

Figure 15. Prediction results from the fully-connected 1-hidden layer MLP models
Model 1: Training without interpolation pre-processing (orange)
Model 2: Training with linear interpolated data (purple)
Model 3: Training with cubic spline interpolated data (red)

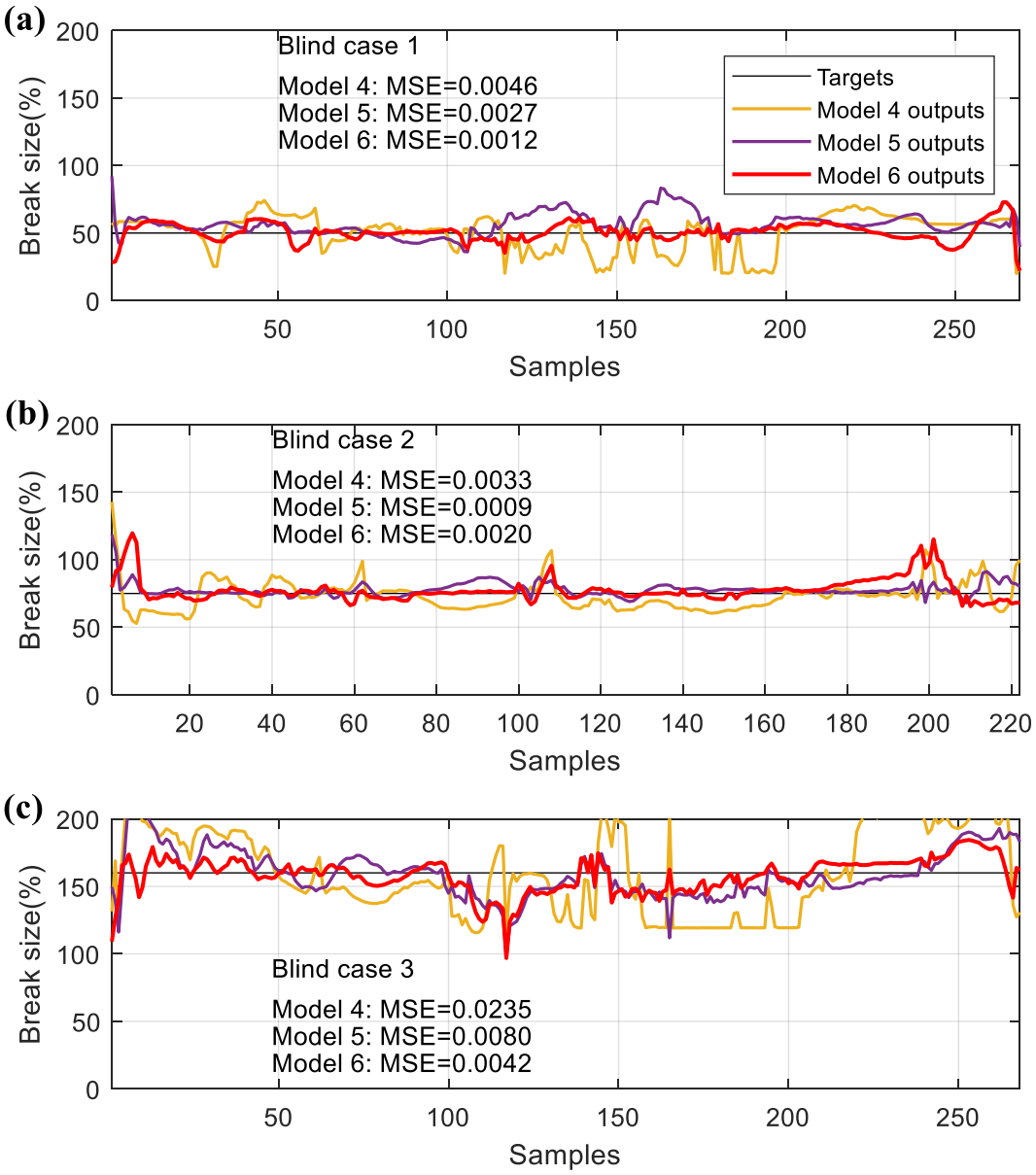

Figure 16. Prediction results from the pruned 1-hidden layer MLP models
Model 4: Training without interpolation pre-processing (orange)
Model 5: Training with linear interpolated data (purple)
Model 6: Training with cubic spline interpolated data (red)

Figure 15 shows the prediction results from the fully-connected 1-hidden layer MLP models (model 1 to 3) with three different pre-processing methods. Figure 15(a) compares the outputs for the blind case 1, which has a break size of 50%. All the outputs of the three models are around 50%, which means that all of the three models can predict the break size of the blind case 1, but the outputs from Model 1 and model 2 have more fluctuation than Model 3, especially for the samples 100 to 200. Therefore, model 3, which applies the cubic spline interpolation pre-processing approach, is more accurate for predicting blind case 1 than the other two models, as it has the smallest MSE value. Figure 15(b) compares the outputs for the blind case 2, which has a break size of 75%. The outputs of the three models are relatively close to the targets, which means they can predict the blind case 2 correctly.

Model 2 and model 3 has similar MSE values which are smaller than that of model 1, which shows that the models trained with the interpolation pre-processing approach can predict blind case 2 more accurately than the model trained without interpolation pre-processing. Figure 15(c) compares the outputs for blind case 3, which has a break size of 160%. The outputs from the three models exhibit significant fluctuation with blind case 3. Model 2 and 3 have smaller MSE values than model 1.

From the comparison in Figure 15, it can be concluded that the fully-connected 1-hidden layer MLP models trained with interpolation pre-processing approach produced more accurate prediction results for the blind cases than the models trained without interpolation pre-processing.

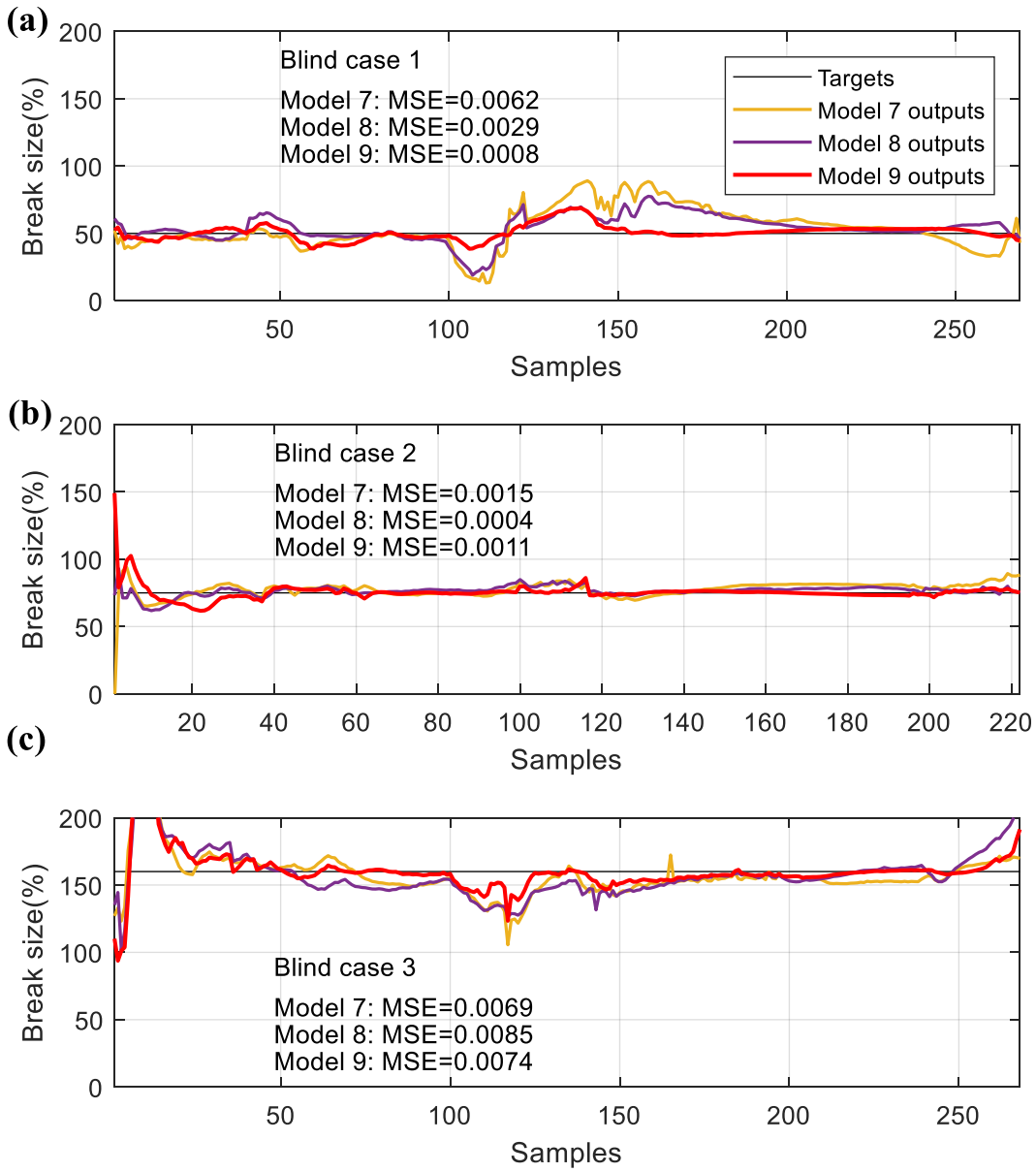

Figure 17. Prediction results from the 2-hidden layer MLP models
Model 7: Training without interpolation pre-processing (orange)
Model 8: Training with linear interpolated data (purple)
Model 9: Training with cubic spline interpolated data (red)

Figure 16 shows the prediction results from the pruned 1-hidden layer MLP models (model 4 to 6) with three different pre-processing methods. The three models can predict the break size for blind cases 1 and 2 with good accuracy, while there is significant fluctuation in the predictions for blind case 3, which is similar to the case of the fully-connected 1-hidden layer models. It is apparent that the predictions of model 4 and 5, which are trained without interpolation pre-processing and linear interpolation pre-processing, respectively, have been improved significantly for blind case 1 and exhibit less fluctuation, compared to the fully-connected case. The pruned 1-hidden layer MLP models trained with interpolation pre-processing yield more accurate predictions for the blind cases than the corresponding models trained without interpolation pre-processing.

Figure 17 shows the prediction results from the blind cases from the 2-hidden layer MLP models (model 7 to 9) with three different pre-processing methods. The outputs from the three models are relatively smooth compared to those from the fully-connected and pruned 1-hidden layer MLP models. The outputs have a little fluctuation in the middle for blind case 1 and at the beginning for blind case 2. In particular, there are large peaks in the first 20 samples of the outputs for blind case 3, which makes the MSE values much higher than those for blind case 1 and 2. Comparing the MSE values of the three models for the blind cases 1 and 2, models 8 and 9, which were trained with interpolation pre-processing, yield lower MSE values than model 7, which was trained without interpolation pre-processing. However, for blind case 3, the models trained with interpolation pre-processing have slightly higher MSE values than the model trained without interpolation pre-processing.

Figure 18 shows the prediction results from the GMDH models (Models 10 to 12) with three different pre-processing methods. The outputs of GMDH models can predict the break sizes of blind cases but their accuracy is not as good as the other architectures considered here because their outputs exhibit much larger fluctuation. Model 12, which was trained using both original data and cubic spline interpolated data, has the smallest MSE out of the three blind cases, compared to models 10 and 11, and the corresponding mean MSE is the smallest as well.
A comparison of the twelve neural network models using MSE values is shown in Table 7. The MSE value for each blind case and the mean MSE value for the three blind cases, which indicates the overall prediction accuracy, are shown for the performance comparison. Comparing the overall prediction performance, the model trained with cubic spline pre-processing has smaller mean MSE values for each architecture, which means they have better prediction accuracy. This is consistent with the simulation study presented in Section 4. Therefore, four models trained with cubic spline pre-processing (models 3, 6, 9 and 12) are selected for further prediction performance comparisons. Although both the fully-connected and pruned 1-hidden networks yielded the best overall prediction accuracy during the blind case validation, using the parsimony principle the pruned 1-hidden network model is to be preferred as it is simpler than the fully-connected one. Therefore, model 3 is kept for further processing.

The mean MSE value of model 6 is slightly smaller than the mean MSE of models 9 and 12. Therefore, the pruned 1-hidden network model (model 6) has the best overall prediction accuracy out of the four architectures. Comparing the prediction accuracy of those three models for each blind case, model 9 has smaller MSE values for two smaller break sizes (which are 50% and 75%) than the other two models by a factor of around 0.5. But for the large break size (which is 160%), model 6 and model 12 have much smaller MSE values than model 9 by a factor of about 0.55, which influence the mean MSE of the three blind cases.

In conclusion, the pruned 1-hidden network model (model 6) has the best overall prediction accuracy, the 2-hidden layer MLP (model 9) has the best prediction accuracy for smaller break sizes (blind cases 1 and 2), and the other two models (models 6 and 12) have the best prediction accuracy for the large break size (blind case 3).

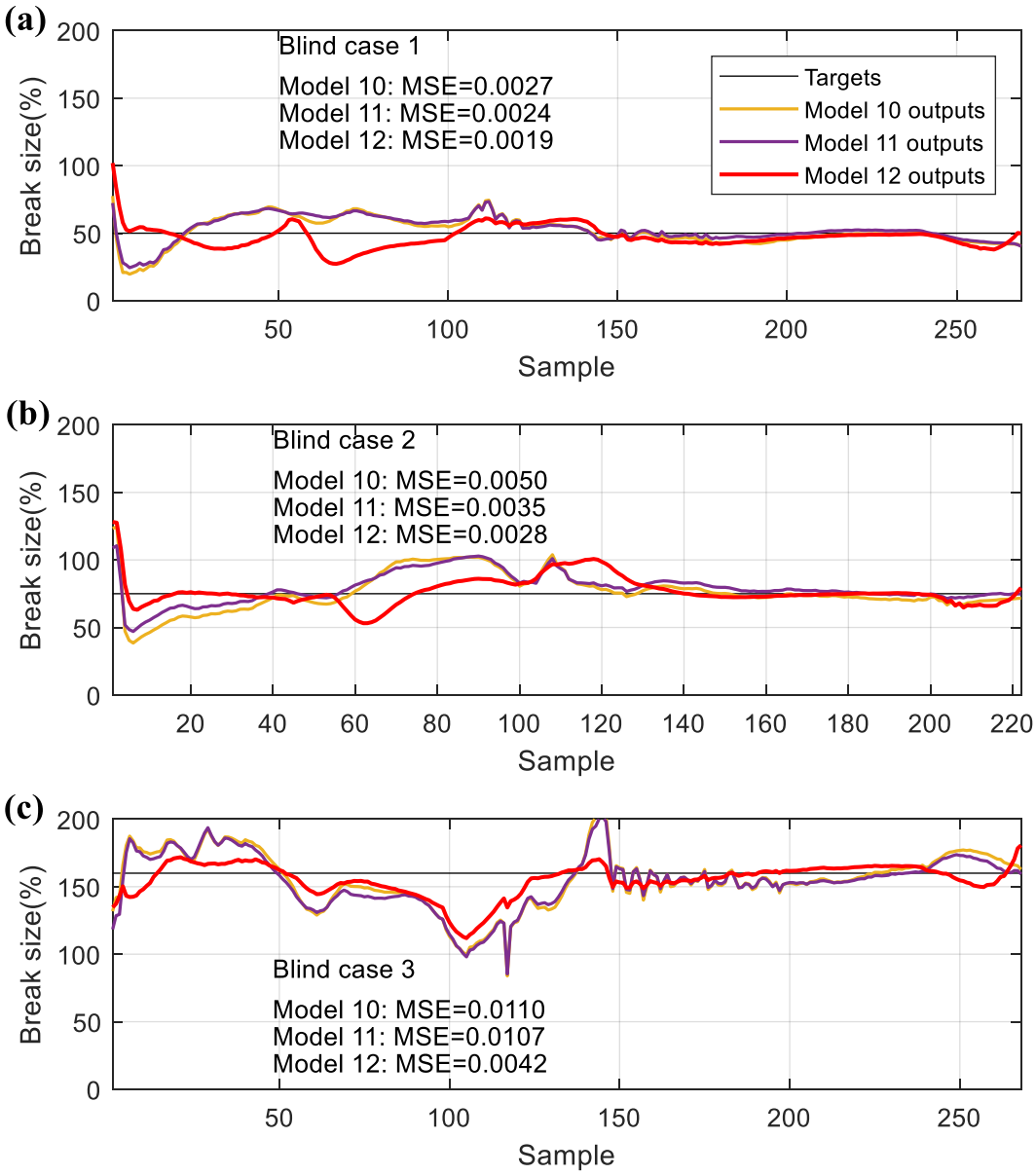

Figure 18. Prediction results from the GMDH models
Model 10: Training without interpolation pre-processing (orange)
Model 11: Training with linear interpolation processing (purple)
Model 12: Training with cubic spline interpolation processing (red)

As adequate blind cases have been used to test the ability of the neural networks models to predict the break size over the whole transient for break sizes not present in the training data set (this strongly tests generalisation ability), we are confident that the best performing models robustly predict the break size even for break levels that are not present in the training data set. Moreover, as neural networks have well known robustness properties, we would expect the predictions to be robust to small changes in certain operating parameters. It should be noted that the neural network models developed in this study are for a 220MWe PHWR at full power operation, which is a common operating mode. However, these models will not be suitable for significantly different power levels at the same plant, such as low power levels and shutdown conditions. Additional models will need to be trained for those conditions.

Table 7. Comparison the prediction performance of all the models based on blind cases

Model No.	Architecture	Interpolation pre-processing	MSE			Mean MSE
			Blind case 1	Blind case 2	Blind case 3	
1	Fully-connected 1-hidden layer MLP	No	0.0069	0.0036	0.0213	0.0106
2		Linear	0.0075	0.0019	0.0060	0.0051
3		Cubic spline	0.0014	0.0021	0.0041	0.0025
4	Pruned 1-hidden layer MLP	No	0.0046	0.0033	0.0235	0.0105
5		Linear	0.0027	0.0009	0.0080	0.0039
6		Cubic spline	**0.0012**	**0.0020**	**0.0042**	**0.0025**
7	2-hidden layer MLP	No	0.0062	0.0015	0.0069	0.0048
8		Linear	0.0029	0.0004	0.0085	0.0039
9		Cubic spline	**0.0008**	**0.0011**	**0.0074**	**0.0031**
10	GMDH	No	0.0027	0.0050	0.0110	0.0062
11		Linear	0.0024	0.0035	0.0107	0.0055
12		Cubic spline	**0.0019**	**0.0028**	**0.0042**	**0.0030**

5.2 A combined diagnostic model

From the simulation study in section 4, the GMDH model trained with cubic spline pre-processing, which corresponds to model 12, has the best consistency. From the blind case validation in previous subsection, the GMDH models also have good prediction accuracy for the blind cases. From the blind case validation results, the pruned 1-hidden layer MLP trained with cubic spline pre-processing (model 6) has the best overall prediction performance, the 2-hidden layer MLP trained with cubic spline pre-processing (model 9) has the best prediction accuracy for small break sizes, while model 6 and model 12 exhibit high prediction accuracy for the large break size. Attempting to improve the results obtained with individual models, we propose a combined model that integrates these three models together to obtain higher prediction accuracy and good robustness. The schematic diagram of the combined model is shown in Figure 19.

The output of the combined model is calculated as a convex combination of the outputs of the three models:

$$y = w_6 y_6 + w_9 y_9 + w_{12} y_{12} \tag{23}$$

such that

$$w_6 + w_9 + w_{12} = 1 \tag{24}$$

where y is the output of the combined model, y_6 is the output of model 6, y_9 is the output of model 9, y_{12} is the output of model 12, and w_6, w_9 and w_{12} are the break size dependent weights, as discussed below. Given that the model outputs are restricted to the range [0, 200%], the convex combination ensures that the combined model output remains within the same range. Note that the range of the weights is [0, 1].

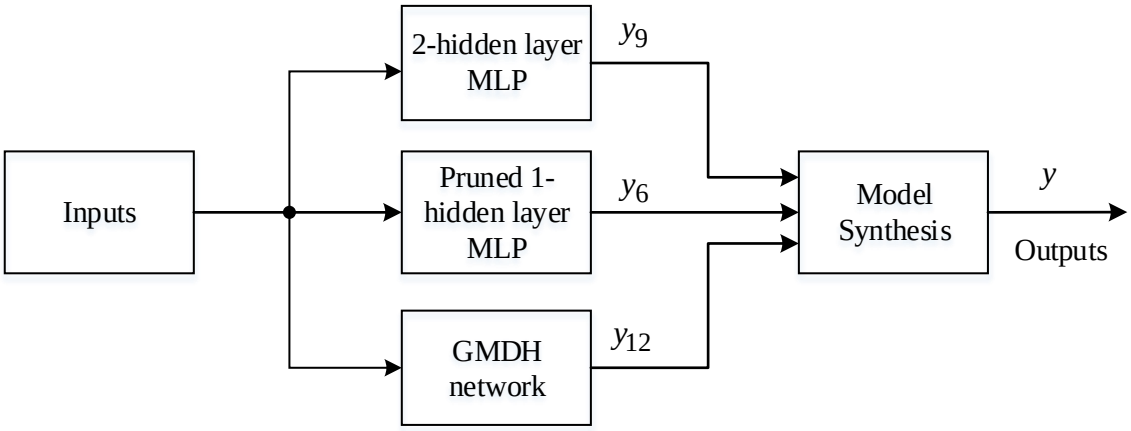

Figure 19. Schematic diagram of the combined model

The weight of model 9 is chosen to have large values for small break sizes and small values for large break sizes. A linear function is chosen to represent the desired weight dependency:

$$w_9 = 1 - \frac{y_9}{200} \tag{25}$$

The weights of model 6 and model 12 are chosen to have small values for small break sizes and large values for large break sizes. Given the constraint (23), w_6 and w_{12} are calculated as follows:

$$w_6 = w_{12} = (1 - w_9)/2 \tag{26}$$

The steps for calculating the output of the combined model are as follows:

(1) Calculate the outputs of the three diagnostic models y_6, y_9, and y_{12}.

(2) Limit the outputs range of each model: if $y_i < 0$, set $y_i = 0$ and if $y_i > 200$, set $y_i = 200$; i= 6, 9, 12.

(3) Determine weight w_9 using Equation (25), and the weights w_6, w_{12} using Equation (26)

(4) Calculate the output of the combined model y using Equation (23).

Figure 20 shows the comparison of the results from the combined model with those from the three selected models. It is apparent that the outputs of the combined model for each blind case are smoother than those of any individual model, which yields smaller MSE values.

The prediction accuracy of the combined model using the MSE values for the three blind cases is shown in Table 8. Comparing the MSE values of the combined model to those of models 6, 9 and 12 given in Table 7, we can see that the MSE values of the combined model for the three blind cases are smaller than the corresponding values for of the three models. The mean MSE value of the combined model is smaller than that of model 6, which has the best mean MSE out of models 6, 9 and 12, by a factor of 0.56. Therefore, the combined model yields very good prediction accuracy for all the blind cases.

Although other ways of selecting the weights could be envisaged, it is clear that this type of convex combination can improve the results when compared with the individual models.

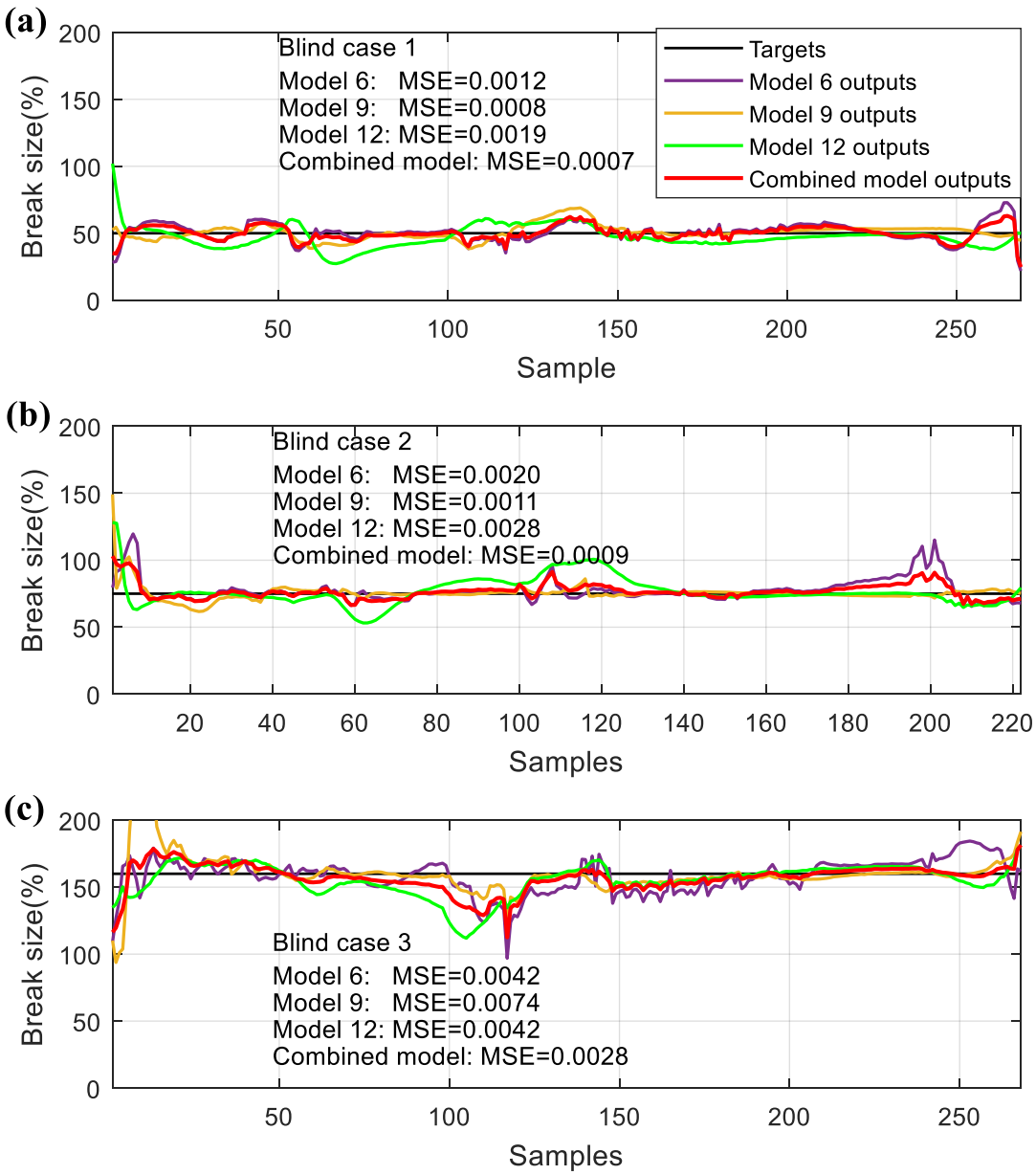

Figure 20. Comparison of the results from the combined model with those from the three selected models
Model 6: OBS pruned 1-hidden layer MLP with cubic spline interpolation pre-processing (purple)
Model 9: 2-hidden layer MLP with cubic spline interpolation pre-processing (orange)
Model 12: GMDH network with cubic spline interpolation pre-processing (cyan)
The combined model (red)

Table 8. Prediction performance of the combined diagnostics model

Architecture	**MSE**			**Mean MSE**
	Blind case 1	**Blind case 2**	**Blind case 3**	
Combined model	**0.0007**	**0.0009**	**0.0028**	**0.0014**

6. Conclusions

This paper has presented a study on the robustness of diagnostic models for predicting the break size in loss of coolant accidents in nuclear power plants. A robustness measure inspired by the leave-one-out cross-validation method has been applied to compare the robustness and consistency of different neural network architectures. Moreover, to deal with the difficulty that the training data is only available for a limited number of break sizes, an interpolation pre-processing method is introduced to generate the additional approximated data for additional break sizes. The robustness and prediction performance of four types of architectures have been investigated using simulated data described in (Santosh et al., 2009), with different pre-processing approaches for each architecture. The simulation study shows that the interpolation pre-processing approach is effective in improving the robustness of a neural network based LOCA break size predictions. The 2-hidden layer MLPs trained with cubic spline interpolation pre-processing exhibited the best robustness, while the GMDH models trained with cubic spline interpolation pre-processing exhibited the best consistency and good prediction accuracy for blind cases. Furthermore, the models were validated with blind cases, and the results indicated that the models trained with interpolated data have better prediction accuracy than those without interpolation pre-processing when using the same architecture, which is consistent with the robustness study. Finally, a combined diagnostics model based on three of the best performing models studied is proposed, and the combined model has been shown to have better prediction accuracy than any of the individual diagnostics models studied in this work.

Acknowledgement

This work was supported by the UK's Engineering and Physical Sciences Research Council (EPSRC) [Grant number EP/M018709/1].

References

Bartal, Y., Lin, J., & Uhrig, R., 1995. Nuclear Power Plant Transient Diagnostics Using Artificial Neural Networks That Allow "Don't-Know" Classifications. Nuclear Technology, 110(3), 436-449.

Choi, G., Yoo, K., Back, J., & Na, M., 2017. Estimation of LOCA Break Size Using Cascaded Fuzzy Neural Networks. Nuclear Engineering and Technology, 49(3), 495-503.

da Costa, R., Mol, A., de Carvalho, P., & Lapa, C., 2011. An efficient Neuro-Fuzzy approach to nuclear power plant transient identification. Annals of Nuclear Energy, 38(6), 1418-1426.

Davis, P., 2014. Interpolation and approximation. Mineola, NY: Dover Publications.

Fletcher, G. D. & Schultz R. R., 1995. RELAP5/MOD3.2 Code manual, Idaho: Idaho National Engineering Laboratory.

G. Vinod, S., Babar, A., Kushwaha, H., & Venkat Raj, V., 2003. Symptom based diagnostic system for nuclear power plant operations using artificial neural networks. Reliability Engineering and System Safety, 82(1), 33-40.

Hagan, M., & Menhaj, M., 1994. Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks, 5(6), 989-993.

Hassibi, B., & Stork, D., 1993. Optimal Brain Surgeon and general network pruning. In IEEE International Conference on Neural Networks. San Francisco, CA, USA: IEEE, 293–299.

Haykin, S., 1999. Neural networks: a comprehensive foundation. London: Prentice-Hall International.

Laurene, V., 1994. Fundamentals of Neural Networks: Architectures, Algorithms and Applications. Upper Saddle River, New Jersey, USA: Prentice-Hall.

Lee, S., No, Y., Na, M., Ahn, K., & Park, S., 2011. Diagnostics of Loss of Coolant Accidents Using SVC and GMDH Models. IEEE Transactions on Nuclear Science, 58(1), 267-276.

Lin, J., Bartal, Y., & Uhrig, R., 1995. Predicting the Severity of Nuclear Power Plant Transients Using Nearest Neighbors Modeling Optimized by Genetic Algorithms on a Parallel Computer. Nuclear Technology, 111(1), 46-62.

Loan, C., 1999. Introduction to Scientific Computing: A Matrix-Vector Approach Using MATLAB (2nd ed.). Upper Saddle River, New Jersey: Prentice Hall.

Lu, B., & Upadhyaya, B., 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.

Mo, K., Lee, S., & Seong, P., 2007. A dynamic neural network aggregation model for transient diagnosis in nuclear power plants. Progress in Nuclear Energy, 49(3), 262-272.

Moshkbar-Bakhshayesh, K., & Ghofrani, M., 2013. Transient identification in nuclear power plants: A review. Progress in Nuclear Energy, 67, 23-32.

Na, M., Shin, S., Lee, S., Jung, D., Kim, S., Jeong, J., & Lee, B., 2004. Prediction of Major Transient Scenarios for Severe Accidents of Nuclear Power Plants. IEEE Transactions on Nuclear Science, 51(2), 313-321.

Na, M., Shin, S., Lee, S., Jung, D., Kim, S., Jeong, J., & Lee, B., 2004. Prediction of Major Transient Scenarios for Severe Accidents of Nuclear Power Plants. IEEE Transactions on Nuclear Science, 51(2), 313-321.

Onwubolu, G., 2015. GMDH-Methodology and Implementation in Matlab. London: Imperial College Press.

Reich, Y., & Barai, S., 1999. Evaluating machine learning models for engineering problems. Artificial Intelligence in Engineering, 13(3), 257-272.

Renders, J., Goosens, A., de Viron, F., & De Vlaminck, M., 1995. A prototype neural network to perform early warning in nuclear power plant. Fuzzy Sets and Systems, 74(1), 139-151.

Santosh, T. V., Vinod, G., Saraf, R. K., Ghosh, A. K., & Kushwaha, H. S., 2007. Application of artificial neural networks to nuclear power plant transient diagnosis. Reliability Engineering & System Safety, 92 (10), 1468-1472

Santosh, T. V., Srivastava, A., Sanyasi Rao, V., Ghosh, A., & Kushwaha, H., 2009. Diagnostic system for identification of accident scenarios in nuclear power plants using artificial neural networks. Reliability Engineering and System Safety, 94(3), 759-762.

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.

Santhosh, T. V., Vinod, G., Ghosh, A. K., & Fernandes, B. G., 2018. An approach for reliability prediction of instrumentation & control cables by artificial neural networks and Weibull theory for probabilistic safety assessment of NPPs. Reliability Engineering & System Safety, 170(1), 31-44.

Su G., Morita K., Fukuda K., Pidduck, M., Jia, D., et al., 2003. Analysis of the critical heat flux in round vertical tubes under low pressure and flow oscillation conditions. Applications of artificial neural network. Nuclear Engineering & Design, 220(1), 17-35.

Tian, X., Becerra, V., Bausch, N., Vinod, G., & Santhosh, T., 2017. A method for measuring the robustness of diagnostic models for predicting the break size during LOCA. In Annual Conference of the Prognostics and Health Management Society. Tampa, FL, USA.

Wang, K., 2018. A study of cubic spline interpolation. Insight: Rivier Academic Journal, 9(2), 1-15.

Zhang, J., Chen, R. H., Wang, M. J., Tian, W. X., & Su, G. H., et al., 2017. Prediction of LBB leakage for various conditions by genetic neural network and genetic algorithms. Nuclear Engineering and Design, 325, 33-43.