




Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems

Ahmad Azharuddin Azhari Mohd Amiruddin¹ · Haslinda Zabiri¹  · Syed Ali Ammar Taqvi^{1,2} · Lemma Dendena Tufa¹

Received: 15 February 2018 / Accepted: 19 November 2018 / Published online: 13 December 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract

The use of artificial neural networks (ANN) in fault detection analysis is widespread. This paper aims to provide an overview on its application in the field of fault identification and diagnosis (FID), as well as the guiding elements behind their successful implementations in engineering-related applications. In most of the reviewed studies, the ANN architecture of choice for FID problem-solving is the multilayer perceptron (MLP). This is likely due to its simplicity, flexibility, and established usage. Its use managed to find footing in a variety of fields in engineering very early on, even before the technology was as polished as it is today. Recurrent neural networks, while having overall stronger potential for solving dynamic problems, are only suggested for use after a simpler implementation in MLP was attempted. Across various ANN applications in FID, it is observed that preprocessing of the inputs is extremely important in obtaining the proper features for use in training the network, particularly when signal analysis is involved. Normalization is practically a standard for ANN use, and likely many other decision-based learning methods due to its ease of use and high impact on speed of convergence. A simple demonstration of ANN's ease of use in solving a unique FID problem was also shown.

Keywords Artificial neural network · Fault detection · Fault diagnosis · Engineering application · Data preprocessing

1 Introduction

Faults in the process industries, equipment, manufacturing industries, power generation, etc., can result in product degradation, increased operating costs, increase in shut-down hours, and environmental degradation [8–10]. However, prompt detection and diagnosis of anomalies can be vital to ensure loss prevention and financial recuperation. As specific economic requirements and environmental demands grow more and more stringent, it became quite clear that companies will require more investment in reducing the occurrence of faults that may lead to instabilities, resulting in tragic economic and safety implications. The term fault detection can be used to refer to both

the detection of causes based on system changes and component fault. In a broad sense, a fault can be defined as undesired changes that can lead to degradation of the overall system performance. Tolerable fault ranges do exist; however, if it is not dealt with for too long, they may convert into a serious fault which can lead to the major breakdown of a system component or function.

The failure of a system can be caused by a wide variety of problems. Typically, the solutions to the problems are known by empirical studies or first principle derivation. In such cases, a detailed procedural solution can prove to be enough when there is sufficient data and knowledge regarding the system behavior [11–13]. In cases where the behavior of a system is either unknown or too complex, the use of artificial neural networks (ANN, or simply NN) is becoming more attractive as an alternative data-based solution. Its ease of use, high noise tolerance [14] and broad applicability for nonlinear systems are only part of the reasons why researchers have explored its use in various general predictive purposes [15, 16] and industrial fields such as transportation [17], agriculture [18],

✉ Haslinda Zabiri
haslindazabiri@petronas.com.my

¹ Department of Chemical Engineering, Universiti Teknologi PETRONAS, 32610 Bandar Seri Iskandar, Perak, Malaysia

² Department of Chemical Engineering, NED University of Engineering & Technology, Karachi 75270, Pakistan

electronics [19–21] medicinal informatics [22], and chemical process industries [23–25]. More specifically in the process engineering industry, a lot of attention in recent years has been paid in the application of ANNs in the adaptive control systems [23, 26–28], modeling and identification of dynamic processes [29–31], and time series prediction problems [32, 33]. A growing interest has been seen for fault detection and diagnosis problem [34–39] using ANNs. As a viable mathematical tool for solving nonlinear problems, the self-learning ability is the main attractive property of neural networks.

While many types of ANNs exist, its basic principles are similar. ANNs can be thought of as universal approximators [40], where the arbitrary relation between two vector spaces can be realized [41]. It is composed of an input layer, the output layer, and hidden layers between them. Connecting each layer are neurons (or nodes) containing weight coefficients that affect the overall structure of the network. Preprocessing (mapping of data into more conditional form) of the input data is a crucial aspect in ANN usage to assist in reducing computational costs [42], lowering burden of requiring many variables, improving generalization of the features within the input space, removal of noise wherever possible, and even as part of the step in obtaining the appropriate features for the input space [2, 15, 43]. Other factors to consider are the choice of training functions (also called “training method”), hidden neuron sizes of each layer, sample size of input space, and testing conditions.

Within the context of engineering and design, the field of fault detection and identification/diagnosis (FID/FDI/FDD) is widely explored [13, 44–47], utilizing various methods, including neural networks [48]. Exploration of ANNs for use in fault detection is not a recent venture. The use of ANN in fault detection has been thoroughly discussed and explored by Patan [23] within the context of locally recurrent ANNs applied in industrial process control. While the monograph discussed a great deal regarding fault detection and ANN fundamentals for use in chemical process systems, it is not obvious in detailing the merits of its use in a broader application.

This paper aims to provide a brief overview of the application of ANN in detecting anomalies or unexpected behavior within various engineering fields. Cases of successful ANN applications for fault detection, with the preprocessing techniques and training methods undertaken (to ensure consistent results), would be of much use to aspiring researchers looking to adopt its use in their analysis as an alternative testing method. Therefore, this review will analyze several methodologies and studies attempting to detect abnormal behavior within engineering and design, covering various disciplines. Section 2 will brief on the nature of ANNs, the most widely used basic ANN

architecture of MLP (multilayer perceptron) and their associated algorithm and methodologies. Section 3 will go over some of the preprocessing techniques used to prepare the input space to be used in ANNs. Section 4 will cover the reviewed applications of ANN for solving anomaly detection and identification problems in Sect. 4.1, with the finer points from all the cases discussed in Sects. 4.2 and 4.3. In Sect. 5, a short tutorial was shown in how ANN is used to tackle a specific problem in a simple manner. Within the context of this paper, “faults” refers to any known or unknown deviation within a system, and the erratic behavior can be observed qualitatively or quantitatively, and is used interchangeably with “anomalies.”

2 Artificial neural networks

A basic representation of a neuron is shown in Fig. 1. The sum of product value, n from the weights, w and inputs, p and bias, b (1) are passed through the activation (or transfer) function, (2) of which the generated output, a will define the transformative characteristic of the particular neuron in the neural network.

$$n = \sum_{i=1}^N p_i w_i + b \tag{1}$$

$$a = f(n) \tag{2}$$

The classification of ANNs is rather broad; however, most are categorized based on the goal of the analysis. They are mainly function approximation, predictive analysis, classification/pattern recognition, clustering, among a few others [49]. The first two relies usually on supervised learning, where the inputs are assigned a target for the iterative structuring of the internal ANN weights. For clustering networks and sometimes classification, it mainly relies on unsupervised learning where inputs are used

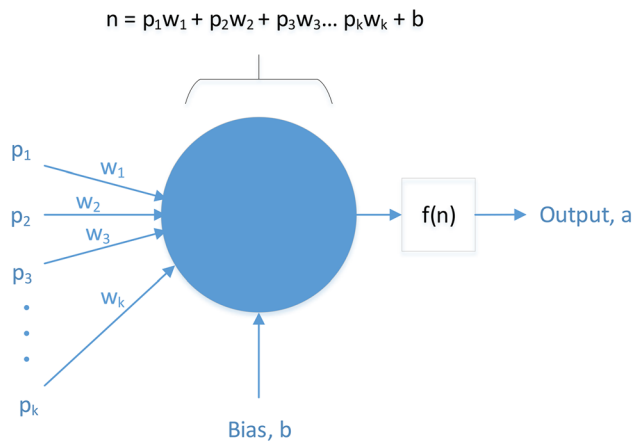


Fig. 1 A single artificial neuron

without targets, and the data will be sorted out by “clusters,” or groups based on certain properties of the inputs themselves, e.g., by shape, by color, or by proximity. Applications and concerns in pattern classification, which are typically the problems manifested in anomaly detection, will be highlighted.

It is important to note that there are limitations behind the usage of NNs. The main weakness is in its inability to generalize properly for a wide range of scenarios [50] unless specifically trained to do so. Even if that is to be considered, the time and effort required to properly establish the various conditions required to be covered for one network alone can be burdensome. Another weakness is in its black-box nature, which limits the understanding of how the decisions are actually made by the network. Lastly, the setup of choosing the right network architecture and configurations, along with the time required in using a large number of data set, can be tedious. The use of NN for solving any major problems simply as an “out-of-the-box” solution is only recommended when the problems are very clearly defined, as the sheer optimization and expertise required to solve a particular problem is still highly tied to the problem in which it is applied to [51]. However, when complemented with other techniques, it proves to be an invaluable tool for gaining insight into nonlinear operations in an exploratory manner, as a means to enable an initial feasibility study for other statistical methods [52], and as a standalone mechanism, if the initial results proved to be incredibly effective.

The standard multilayer perceptron (or feed-forward) neural network (Fig. 2) contains multiple neurons comprising hidden layers and hidden sizes/units. A higher number of layers will compensate for the weakness of the

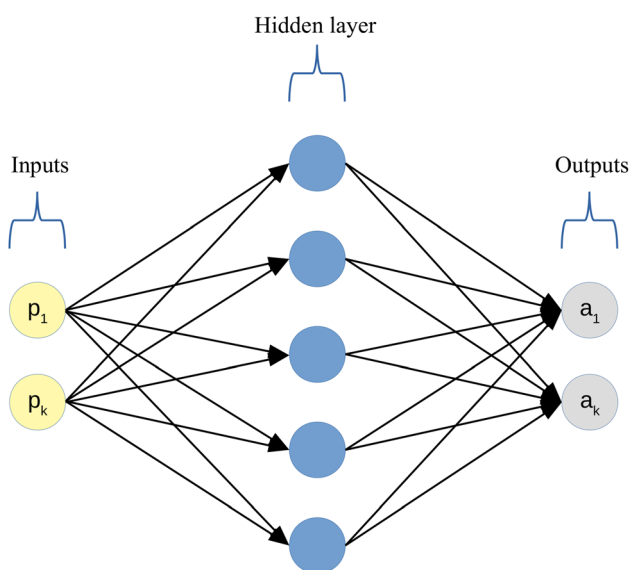


Fig. 2 Multilayer perceptron neural network

perceptron’s linear nature and allow for nonlinear calculations to be mapped [53]. An optimal hidden size/unit will give better performance, though the optimal number for each system must be obtained through approximation, trial and error, or heuristic methods. The user will usually find an ideal value rather fast by starting from 10, and testing higher or lower values.

A simple way to describe its setup and usage is that it is essentially the same as functional analysis, whereby you have a large sample set of data, each set with a range of independent parameters and dependent parameters. The entire set is termed as “inputs,” with the independent parameter (the “y” of the equation) termed as targets. The output refers to the generated values from the network. The supervised learning can take place in an offline/batch manner, where the network is trained sequentially, across the entire sample set. Another method is through online learning, where the long-term memory is involved, and information decay is a necessary factor. The latter method will not be explored in depth, however.

A large reason behind why MLP networks are highly preferred for function approximation tasks is due to its various back-propagation (BP) methods and optimized forms which are very efficient in seeking convergence. The basic BP algorithm in its earlier forms was very slow to converge for larger networks, though this problem is eliminated by the use of learning parameters in weight adjustment and higher-order computation schemes. In the discussion of loss functions (also known as cost/objective functions) to use, it is dependent on the studied system, however, not so much in FID as most problems are easily reducible with the use of mean square error (MSE) as the iterative means for gradient changes. Special considerations are mostly for unique types of data such as pattern recognition-based inputs.

It is important to note that loss functions are not necessarily the same as the performance metrics used to evaluate the network. For curve fitting applications, MSE is used both to calculate the weight updates for each training step while also quantitatively showing how far off the average prediction errors are in the network behavior as a performance metric. However for classification problems, an exercise of judgment is needed for the choice of performance metric. While the MSE can be used as the loss function to train the network behavior, it will only give you the real-value functions of your outputs (e.g., for an arbitrary system where the vector output of [1, 0] is a value designated with the “good” class while [0, 1] is designated as “bad”). The users will have to also determine their own threshold values so that the generated value can be qualitatively categorized and become statistically interpretable, usually in the basic form of true/false accuracies (e.g., from an output value of [− 0.1, 0.8], would you interpret it as a

“good” class?). Using the same example, assume 20 random samples of supposedly “good” data set and another 20 set of “bad” classes of data are fed into a prior-trained network. Out of the 20 “good” sets, 18 of them predicted a value close to [1, 0] instead, while 15 of the “bad” sets were classified with values around [0, 1] value. At a glance, it manages to predict the “good” behavior at 90% accuracy, while it predicted “bad” behavior only at 75% accuracy. It is also very normal for some of the misclassified values to have seemingly sporadically generated values such as [0.5, 0.32]. At this point, it is entirely dependent on the experimenter to consider the next action. Are the levels of accuracy acceptable? Is this result acceptable/unacceptable given the sample size? Is true/false classification the only important metric (how significant are true/false positives to the system)? Should more preprocessing methods be explored to ensure better behavior is recognized by the ANN? Are the architecture and other ANN heuristics properly selected?

Further developed methods such as variable-learning rate BP and resilient BP that alter the learning parameters in an iterative manner are some of the answers to that problem. From a practical standpoint, a method most preferred for function approximation is the Levenberg–Marquardt BP (LMBP) and for pattern recognition tasks the scaled conjugate BP [54]. Other supervised training methods are radial basis functions (RBF), recurrent networks, and Hopfield networks. While the speed of convergence of the LMBP is between 16 and 136 times faster than the standard conjugate BP [55], the selection of the appropriate training algorithm is still primarily based on the type of task required. Though the obtained network is a black-box model, efforts in converting the weighted neurons into a usable white-box model through rule-based pruning and extraction of the weights have been done [56]. Similarly, if the exact weights within the model are known, the relative significance of each input parameter in influencing a particular output can be analyzed through a formulation by Garson [57]. To evaluate the neural network performance in replicating the intended system behavior when confronted with foreign input data, a general method used in training the data set is to split the original data set into multiple sets of data at a specified ratio, the majority for use in training the network, the remaining for verification and fine tuning of network, and for testing its performance against “foreign data,” respectively.

Among the NN architectures, recurrent NNs (also referred to as dynamic neural networks, DNN) are a rather viable option to use as it has additional learning capabilities due to their inclusion of dynamically driven feedback [58]. Unlike a purely static multilayer perceptron, recurrent NNs utilize not only the input in predicting a particular output instance, but also any state in between (most

commonly the output) to contribute in predicting the next output value. The most common form of which is the nonlinear autoregressive networks with exogenous inputs (NARX), which uses the output to influence the input in a feedback loop. This allows for the learning process to account for temporal changes in the input and output as well, which makes it an attractive option for dynamic (time-wise) data set modeling [58]. The inclusion of a feedback also necessitates new training methods depending on the architecture, such as back propagation through time (BPTT) and real-time recurrent learning (RTRL). While recurrent NNs has its own set of challenges such as Levenberg–Marquardt have been introduced to alleviate the problem and difficulties retaining long-term information in larger time scales [59], the use of detailed nonlinear sequential estimation algorithms or second-order optimization techniques such as Levenberg–Marquardt. Additional properties of recurrent NNs such as tapped delays (which alters the degree of delay from the feedback to the input, in the form of a memory state) and placement of feedback lines are present for consideration in addition to the standard NN heuristics. The authors would like to disclaim that the cases studied do not involve any use of NARX. Even in cases where recurrent NNs are involved, the heuristics and properties involved are not touched upon.

3 Preprocessing of inputs

Preprocessing refers to the conditioning of the input space prior to using it for training of a network. While the proper selection of NN architecture is crucial for a network to properly model the behavior of a system, it will not necessarily guarantee an accurate training process. The general problems that plague the training of NN are convergence to local minima instead of global minimum, overfitting (no generalization/flexibility of trained model), lack of salient features or variables to represent the model behavior, too small a sample size, among others. As with most simulation-based analyses, the output generated from the network is only as good as the inputs used to create the network.

3.1 Normalization

The most widespread and common technique, the benefits of normalization are most obvious in speeding up convergence rate and improving computational efficiency. A sigmoid activation function is often used for MLP networks and if the net sum product of inputs and weights is greater than 3, the functions essentially become saturated. For example, an output of 3 will adjust the weights of a neuron using a sigmoid function at a much smaller rate compared to a much smaller value of 0.1 ($\exp(-3) \approx 0.05$, against

$\exp(-0.1) \approx 0.905$). This will lead to very small gradient changes, especially if the weights are not small enough to compensate for the large input values [60]. By normalizing the value ranges to sufficiently small values, the values entering the activation functions will operate at a faster pace. Therefore, normalization of the input space is basically a standard for most training procedures, and even as preparation for some preprocessing steps such as principal component analysis. Normalization for a custom range can be done according to (3), where z is the normalized value, T is the top boundary, and B is the bottom boundary. The maximum and minimum values used should be within the entire sample set, inclusive of both training and testing sets. In order to generate the actual output value prior to the normalization step, a denormalization step is typically included within most NN creation algorithms.

$$z = (T - B) \left(\frac{x - \min(x)}{\max(x) - \min(x)} \right) + B \quad (3)$$

While normalization is typically done at the $[-1, 1]$ range or $[0, 1]$ range, the proper range selection should depend on the ranges of the activation function. For example, a hyperbolic tangent activation function should use the $[-1, 1]$ range and a logistic function will use the $[0, 1]$ range. Various other forms of normalization exist, and different techniques have shown to affect performance based on the training method used in the network [42].

3.2 Feature extraction and data transformation

The conditioning of the input space is extremely important in order to allow a more accurate modeling through neural networks. Its efficacy can be noted even through a simple transformation of a data set into natural logarithmic forms [43]. A dilemma that often plagues machine learning is the curse of dimensionality [61], i.e., the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables (i.e., dimensionality) of the function. Therefore, it is recommended for an input to use the features containing the most significant information, in order to ensure that extremely large data requirements are avoided, while at the same time the network is able to learn from the most characteristic inputs of a modeled system. An imbalance between the trade-off can be rather easily be detected in the network through any overfitting or underfitting seen and evaluated in the training and testing performances.

Principal component analysis (PCA) is one of the simpler methods in multivariate analysis that can be used to reduce the number of required parameters of a data set without much loss of information [62]. Assume a sample vector X of n samples, and m parameters, represented in a

matrix such that $X = n$ by m . The dimension reduction is done through forming a covariance matrix of the sample sizes and total parameters, and use the k th component (or principal components) of the eigenvector representing the reduced dimension (such that $k < m$) to represent the original matrix in reduced form of Y . A new reduced data set can be obtained by reversing the process of obtaining the covariance matrix, using the representative eigenvector. Essentially, a lower dimension sample is obtained that can represent the salient quantities of m parameters in a dimension, usually 2 or 3, represented in the form of $Y = n$ by k , that is easier to work with.

Feature extraction, primarily in analysis of signals as input, can be taken from transformation methods such as Fourier transforms (FT) or wavelet packet decomposition (wavelet transforms). The wavelet transform is created as a method to overcome the limitations of the FT, where the entire range of the time-domain signal was used to obtain a frequency-domain sinusoidal plot of the signals. The short time Fourier transform (STFT) was the precursor to the wavelet method, which simply analyzed a specific time window of interest for the frequency values [63]. The wavelet transform, however, also takes into account the shape and size of the regions. Wavelets are functions generated from one single function (basis function) called the mother wavelet by dilations (scaling) and translations (shifts) in time (frequency) domain [64]. Its main purpose is to be able to effectively capture signal characteristics at a higher and more localized resolution [65], allowing for accurate capturing of features for use in neural networks. Aside from that, extraction of additional statistical features from discrete time-based signals or frequency data can be done simply by taking properties of the overall sample such as mean, variance, standard deviation, skewness, kurtosis, and n th-order moments.

3.3 Coding of class targets

Mostly applicable for classification and pattern recognition problems, the manner in which the class matrices are formatted as targets for the input can be assigned in various ways. For example, a problem involving 4 classes can be coded in three ways [60]. Firstly is the assignment of singular values for each input within the sample space, e.g., (1), (2), (3), (4) Secondly, two-dimensional values can be used, in the form of binary classifiers, e.g., (0, 0), (0, 1), (1, 0), (1, 1). Thirdly, very simple n -dimensional values corresponding to the n number of classes can be used in the form of n -ary classifiers, e.g., (0, 0, 0, 1), (0, 0, 1, 0), (0, 1, 0, 0), (1, 0, 0, 0), can be employed for higher class type requirements. The third method is noted to have shown the best results and is usually the standard for a first-try basis [60]. The magnitude of the values can also be used to

represent the degree of “intensity” for the assigned class, i.e., (0.1, 0.1, -0.1, 0.9) shows that the particular set indicates a 90% accuracy of a match to the 4th class, while showing minute signs of relativity to the 1st and 2nd class, and showing anti-correlation to the 3rd class. In this case, this particular set of values can be (depending on how strict the threshold is established) determined as belonging to class 4, for categorical purposes. If the data set shows little to no correlation, the value will be spread across all spaces. Furthermore, for studies where severity of the faults is also involved, the range of values representing the entire fault can be discretized to a custom range as well [10, 66]. This will allow for training of inputs that ascribe different degrees of faultiness for a certain class. It is exactly due to its simplicity that the testing of different code configurations for target classification merits consideration in the preprocessing step of a classification problem.

4 Discussion

Neural networks have historically been used for many purposes, one of them being in FID. Even within the niche of FID applications in engineering, there are a considerable number of cases of successful applications in terms of both early adoption and variety of engineering fields. While this paper is only a brief review of the simpler implementations, the cases covered are found to be from the fields of chemical, electrical, and mechanical engineering. It should be noted that this is due to how broadly the term FID is used for the selection of cases for this review. This section is divided into two main parts. The first part, in Sects. 4.1 and 4.2, is a list of summaries for each studied case of ANN application in FID used in this review. Each case is further summarized in Table 2, based on the main problem solved by the authors, the architecture used, and the observed preprocessing techniques. The second part, in Sect. 4.3, is a critical analysis of the similarities and dissimilarities between all the studied cases based on a few select characteristics of: problem type, input type, output type, and time sensitivity of data (static vs dynamic). Each case and their respective characteristics are summarized in Table 3. The reasoning behind why the characteristics are chosen as such is also described within the second part.

4.1 Case studies: applications of NN in FID

While the topics covered are rather broad, this paper covers mostly NN implementations to solve FID issues within the engineering and design fields. Hwang et al. [7] used an MLP BP network to detect and diagnose faults on a pressurized water reactor type nuclear plant (Fig. 3). No preprocessing was noted, and 3 distinct neural networks were

used, the first for parameter identification, the second for establishing the threshold fault limits, and the third for detection and diagnosis of faulty condition. A guide for the inputs and outputs used in each neural network is shown in Table 1. The first NN is used to determine a nonzero variable in the dynamic model, the moderator temperature coefficient of reactivity α_c . Obtaining the variable is important as it deviates from time compared to other known constants and shares a critical role in determining just how much the system is allowed to deviate from nominal performance. As it varies in time, the 5 input used for the first NN is time plus 4 residual values (difference between actual output and nominal model output) that has the greatest sensitivity in affecting the coefficient. The 4 residual values are from the reactor power δP , precursor concentration δC , fuel temperature δT_f , and steam pressure δP_s (all represented within by $r(t)$ block). The NN was trained with values generated from a simulation, ranging within $\pm 2\%$ of its nominal value. The output generated is the moderator temperature coefficient of reactivity, used to predict an “ideal” model of the process. The second NN is the threshold logic generation network, which establishes the faulty conditions using the determined coefficient α_c for the creation of a new compensated model. This will allow for better accuracy in detecting faults, ensuring that a better defined state space for faulty conditions is established. The 2 inputs used are time and the coefficient α_c , and the output generated is the error threshold values for 8 of the model parameters. The 8 model parameters are reactor power, precursor concentration, fuel temperature, steam pressure δP_s , cooler temperatures in 1st and 2nd node, temperature of primary coolant node δT_p , and tube metal temperature δT_m . The third network is for the fault detection and diagnosis. While the fault magnitude of each failed sensor and the actuator can be obtained from each parameter’s residuals, the feedback loop of the system ultimately causes each residual to affect the steam pressure. Thus, the steam pressure not only has its own residual value, but carries the other parameters’ as well. This leads to the input of the third NN to be steam pressure residual δP_s at 3 different time instances, $\delta P_s(t_k)$, $\delta P_s(t_{k-1})$, and $\delta P_s(t_{k-2})$. The 2 output nodes are associated with the failure condition of the δP_s sensor and the actuator. The two tests were done on the network; the detection scheme was able to successfully determine the magnitude of the residual error and at the same time classify the source of the residual (the sensor belonging to which actual malfunctioning parameter). Despite using a total of 3 networks for the entirety of this fault detection process, it was noted that the FID system was not designed to feedback the information to the controller for automatic correction to take place, though the corrective step did not state any sort of manual action taken either.

Fig. 3 Block diagram of the fault detection and diagnosis scheme used by Hwang et al. [7]

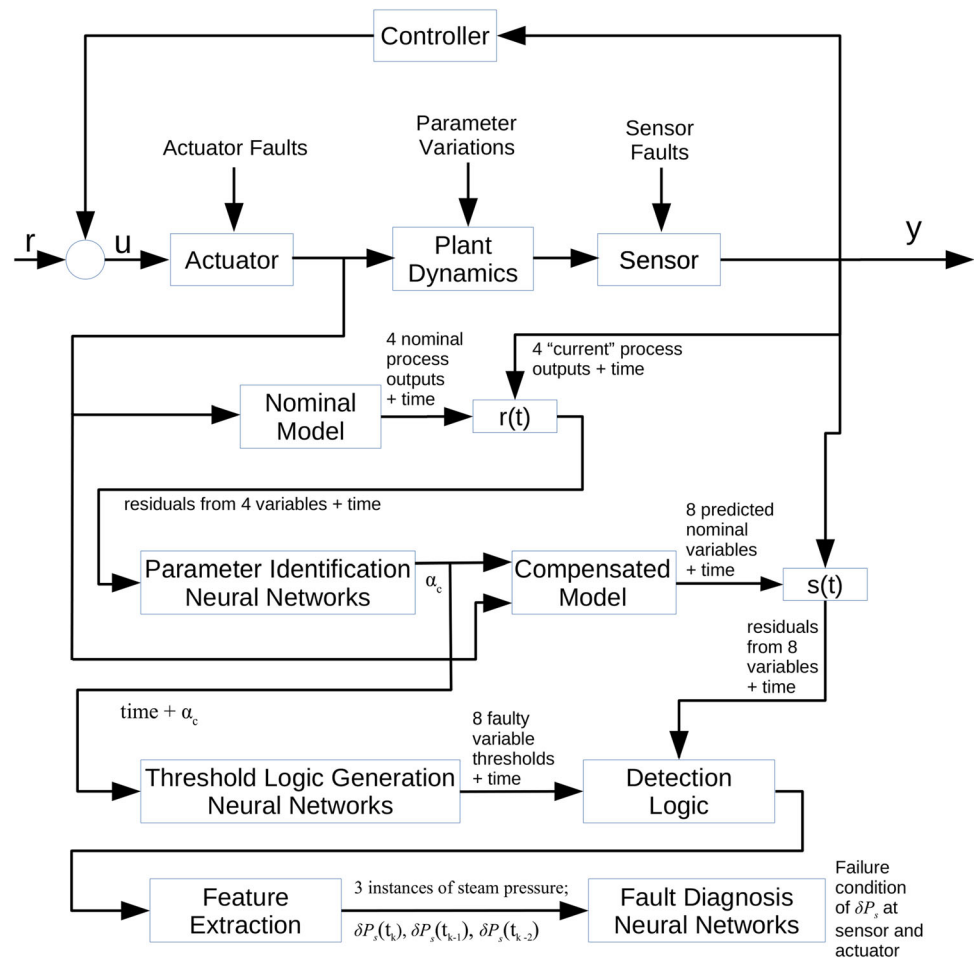


Table 1 Summary of Hwang et al.’s application [7] in terms of neural network activity

First neural network (parameter identification)		Second neural network (error threshold generation)		Third neural network (fault detection)	
Inputs	Outputs	Inputs	Outputs	Inputs	Outputs
Residuals of 1–4 1. Reactor power δP 2. Precursor concentration δC 3. Fuel temperature δT_f 4. Steam pressure δP_s 5. Time	Moderator temperature coefficient of reactivity α_c (to obtain 4 other variables)	1. Time 2. α_c	Threshold ranges of 1. Cooler temperature at first node δT_{c1} 2. Cooler temperature at second node δT_{c2} 3. Primary cooler node temperature, δT_p 4. Tube metal temperature, δT_p 5. δP_s 6. δP 7. δC 8. δT_f	1. $\delta P_s(t_k)$ 2. $\delta P_s(t_{k-1})$ 3. $\delta P_s(t_{k-2})$	Fault conditions of δP_s at 1. Actuator 2. Sensor

Paya et al. [67] used an MLP BP network to detect and classify 6 different vibrational signals from a gearbox. The 6 signals were generated from distinct gear and bearing conditions. Wavelet transforms were used as a

preprocessing step on discrete time-domain signals using the Daubechies D4 mother wavelet, as it was the commonly used wavelet for the time. The network conditions were hard to interpret as the author was using the term

“input nodes” to represent input layer entirely, rather than discussing the features and samples separately. The number of features used was 2, which are wavelet number and amplitude. For each bearing condition, 10 sample sets of the features were used. Six output nodes corresponding to 1 normal bearing conditions and 5 different fault conditions were used. The bearing conditions are: good bearing and faulty blip gear; good bearing and faulty shaved gear; faulty bearing and good gear; faulty bearing and faulty blip gear; and faulty bearing and faulty shaved gear. The cost function used in determining implementation success was mean squared error (MSE), in which the authors indicated the values 5 as “good” and 1 as “excellent” performance. It was noted that the paper had studied the wavelets individually and could not obtain any meaningful classification behind the data alone. The use of wavelets as inputs in the ANN, however, yielded not only very good MSE training results (0.013), but the test performance gave an overall classification rate of 96%.

Aminian and Aminian [20] used a 3-layer MLP BP ANN to determine faulty components within an electronic circuit by analyzing its impulse response. In order to obtain their input data, the analog circuit signals were preprocessed with wavelet decomposition using the Haar mother wavelet, PCA, and normalization. The faults in this case are differences in nominal values of the circuit components. The inputs used are 5 features from the preprocessing, and the outputs are the 9 fault classes assigned to each target sample. The output is considered faulty if the predicted value of the network is not within 0.1 MSE. The authors have noted that the selection of the right wavelet coefficient for the appropriate system can ensure optimal ANN performance; this is likely due to the fact that it is the most important preprocessing step in determining the input space to be used for the system. The results, in comparison with a similar work using ANN [21], have shown similar performances in fault diagnosis performance but achieved with a simpler architecture which utilizes less processing time, due to the aforementioned preprocessing techniques.

Kumar [5] used two types of ANN for the detection of various textile defects, through segmentation of static images and gray-value signal projections. Normalization was used for both networks as a preprocessing step. The first method (Fig. 4) utilizes an MLP LMBP network, with the fault analysis objective of being able to recognize the type of defect through training and testing of image

features. The training of each individual image defect was done separately to see if the a characteristic image was observable for each defect (2 types of weaving styles with 4 fabric defect types each; totaling 8 class of defects). The training was not done with the “defect class” as the output, however. The inputs are simply preprocessed photos of the defects, and the output is an image highlighting the defects in a characteristic pattern. Raw data obtained from the images are characterized using three different pitch sizes, to obtain feature vectors in the form of sub-images to represent any defects through gray-level color variations. The pitches were preprocessed with normalization and histogram equalization, followed by PCA decomposition into smaller dimensions (features). The inputs used range from 6 to 11 features, followed by a median filtering post-process of the output to finalize the feature extraction. The training error (as high as magnitudes of 10^{-7} and as low as 10^{-11}) indicated a successful application for prediction. The author’s exact wording of using “5 nodes in the input layer” (among other values when referring to the size of the input layers) may have been a misinterpretation of terms, as he might actually be referring to the size of the hidden layer to be 5.

A secondary ANN method was also employed using two separate linear feed-forward networks. The input for each neural networks is scanning the raw vibration free images for defects in both 1-dimensional horizontal and vertical signals, preprocessed with normalization (Fig. 5). The objective of this NN is similar, which is to locate defects on the image of a fabric containing a defect and recognize the type of defect. The author hypothesized that by reducing the search space from 2D to 1D, there will be a difference in textile defect detection performance. Data points sampled along the horizontal and vertical spaces of an image of the fabric, in the form of gray-level projection signals, were collected and used as the input. The threshold gray values as a reference for detecting defective signals were calculated with defect-free images. In the testing stage using fabric images with defects, the magnitude of the coloration difference at certain pixels can be seen to indicate defective surfaces on the fabric. Comparing the performance of both ANNs in detecting textile defects, the first method is preferred as it holds the advantage of being able to perform for lower resolution images in detecting defects.

Banjanovic-Mehmedovic et al. [68] used three different ANNs of MLP, Elman recurrent NN, and PNN for the

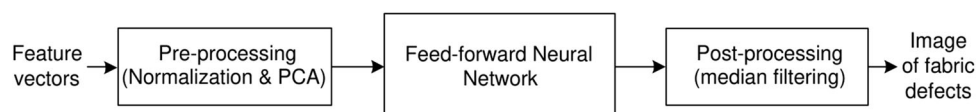


Fig. 4 Flowchart of the feed-forward network used by Kumar (first method of image segmentation) [5]

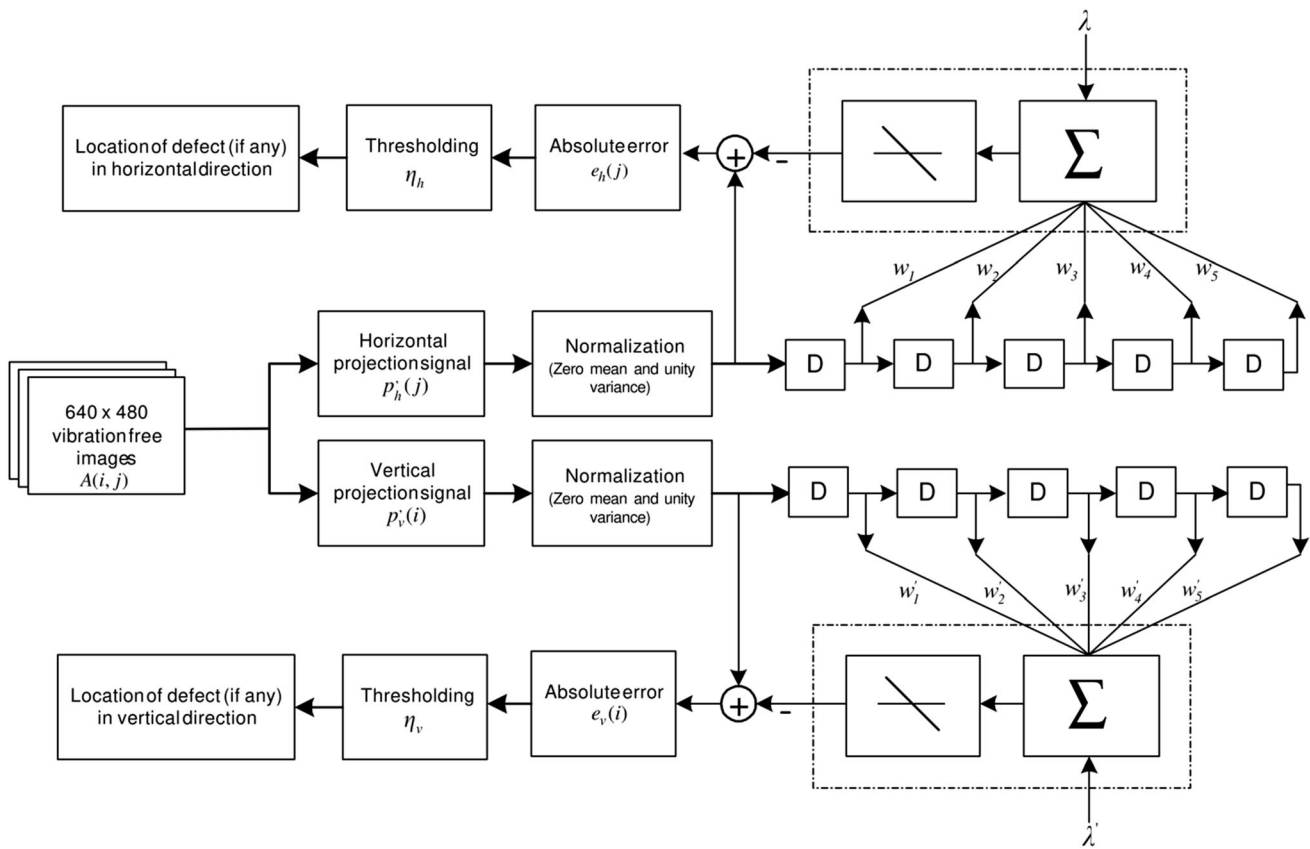


Fig. 5 Diagram of the linear feed-forward network used by Kumar (secondary method of linear color projection) [5]

detection of anomalies, through classification. The class of “anomalous condition” is represented by operating variables not within nominal operating range, while “normal condition” is represented by variables within nominal operating ranges. Two separate networks were created for the anomaly detection of two different equipment sections utilizing different input spaces, which are steam superheaters and steam drums. Six input features were used for each equipment section, each of them representing process variables related to each representative section. Two sets of data per equipment section (one of nominal variable performance, the other of anomalous performance) with sample size of 962 each were used for training and testing. The performance metrics used were accuracy (ACC), true positive rate (TPR), true negative rate (TNR), positive precision (PR), negative precision (NPR), and optimal prediction score (F1). The MLP and Elman recurrent NN were also tested under differing neuron size, learning rate, momentum, and epoch number. The training and testing was done through k -fold cross-validation with $k = 10$, taking precaution in performing the validation multiple times and taking the mean to reduce stochastic variability of the results. The detection performance of the PNN model achieved the best results, showing the total accuracy, PR, NPR, and F1 score of 99.9%.

Samanta et al. [4] investigated the effect of genetic algorithms (GA) on the performance of ANNs for the detection of bearing faults (Fig. 6). It is also noted that the experiment exploited various statistical elements of the generated signal in order to maximize trial choices. Experimental data were prepared from normal and defective bearings, acquiring two sets of 24,576 samples, each segmented into 24 bins of 1024 samples. The signal inputs used were obtained from horizontal and vertical vibrations, which constructed the magnitude parameter z , which also lead to the extraction of the features: mean, RMS, variance, skewness, kurtosis, and normalized fifth to ninth central movements (9 features). The 9 features also provided derivative and integral signals each, totaling the features to 27 (10–27). Another set of features was made from running the 9 initial features into high and low pass filters (28–45), totaling the final feature count to 45 for use in the input vectors. Three different normalization schemes (zero mean ± 1 , zero mean with standard deviation of 1, and zero mean with maximum absolute value) were tested with the 45 features. The ANNs used to train and test the inputs and their 2 bearing conditions assigned as targets (normal and faulty) are feed-forward MLP, RBF, and probabilistic neural network (PNN), all using the MSE as the performance function. In conclusion, the inputs whereby 6

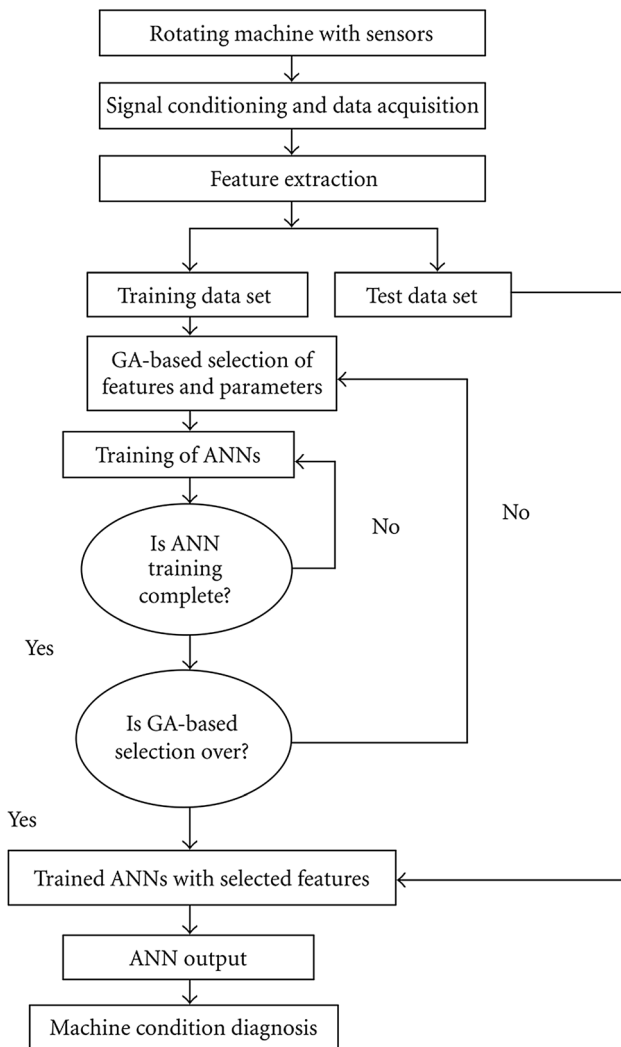


Fig. 6 Flowchart of diagnostic procedure used by Samanta et al. [4]

selected features were chosen from GA gave 100% classification performance for most of the ANN testing. Additionally, even though all ANN gave similar performance results, the training time of PNN was the lowest. The features generated from lower-order statistics (1–9) generally gave better performance; however, the inclusion of higher-order moments was also included in the 6 features selected through GA for use in the ANN, justifying its presence. The classification success of the statistical normalization scheme (with zero mean ± 1 and standard deviation of 1) is slightly better than the magnitude normalization scheme for lower number of features (up to 3). However, the test success deteriorated with the scheme of statistical normalization for higher number of features. Training time increased somewhat with higher number of features. The zero mean with maximum absolute values normalization did not have a noticeable impact in improving the classification performance.

Zhou et al. [2] created a novel multi-step fault detection system (Fig. 7) for identifying the source of faults within erratic parameter readings. The network was shown to be effective when implemented for cardiorespiratory parameters generated from a COSMED K4b² metabolic measurement system. PCA and singular value decomposition were used as the preprocessing step for feature extraction for fault identification and generating the error threshold in such a large parameter system, and an MLP BP NN was used for identifying whether the fault is a temporary transient reading or due to a broken sensor. The original data set consists of an unnumbered sample size, each represented by a huge set of 22 parameters. The number of principal components used is determined as 3, based on CPV (cumulative percent variance) analysis. The principal components obtained from PCA are then divided into groups through *k*-means clustering. From the clustering, two clouds of data showing corresponding patterns were shown to be found. From the two clouds of data, two separate states of faulty and non-faulty were successfully seen for the fault identification system to use as a means of classification. The main criterion for detecting faults is the SPE (square prediction error), a method used for multivariate normal data sets [69]. The features generated from the preprocessing step were shown to be within stable SPE readings in tests. When a fault is detected, indicated by the SPE reading going beyond tolerable limits, a BP NN is used to train the past 10 readings, i.e., $X(t_{k-10})$, where $k = 0, 1, \dots, 10$, an X represents the 22 parameter set. The behavior is then simulated to predict the future behavior of the parameters virtually, forward in time. The type of

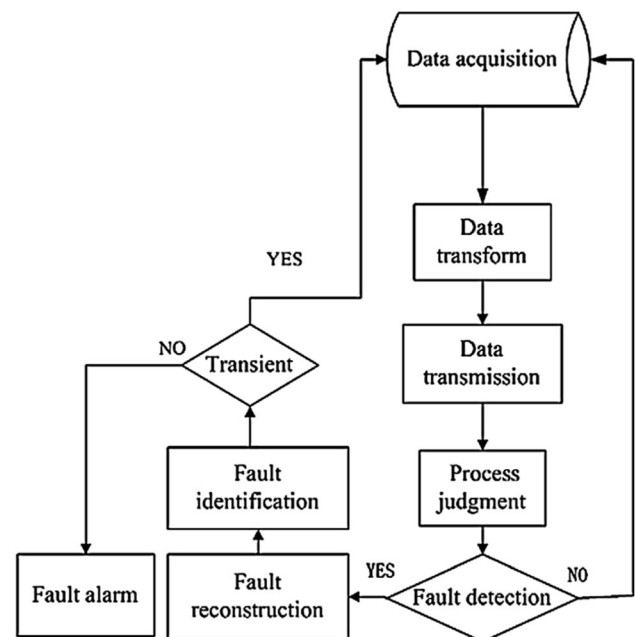


Fig. 7 Multi-step scheme used by Zhou et al. [2]

inputs used was identified as the full 22 parameters represented by the 10 previous time instances, with the output being the expected values for all parameters (except for the anomalous O_2 -related parameters) at the next time instances. If the future patterns only show transient movement, i.e., the pattern does not exceed the tolerable SPE threshold; it will be counted as a transient reading. Should the reading continue to break the SPE threshold, the fault alarm is activated. Furthermore, the variable containing the faulty sensor can be located thanks to any noticeably large error (of the 11th time-sample prediction and the previous 10) among the 22 predicted parameters from the BP NN.

Pandya et al. [1] used an MLP NN to identify and classify different defective conditions of ball bearings based on features extracted from the vibration signal. The preprocessing step (summarized in Fig. 8) involves running the signals through wavelet packet transform of “rbio5.5” family (Reverse biorthogonal 5.5), followed by the extraction of sub-band energy content and kurtoses of the wavelet coefficient for use as features in the input vector. The total input nodes used are based on utilizing the kurtoses and sub band energy as the features taken from the first 3 key coefficients of decomposition, both for horizontal and vertical signals (totaling $2 * 3 * 2 = 12$ input nodes).

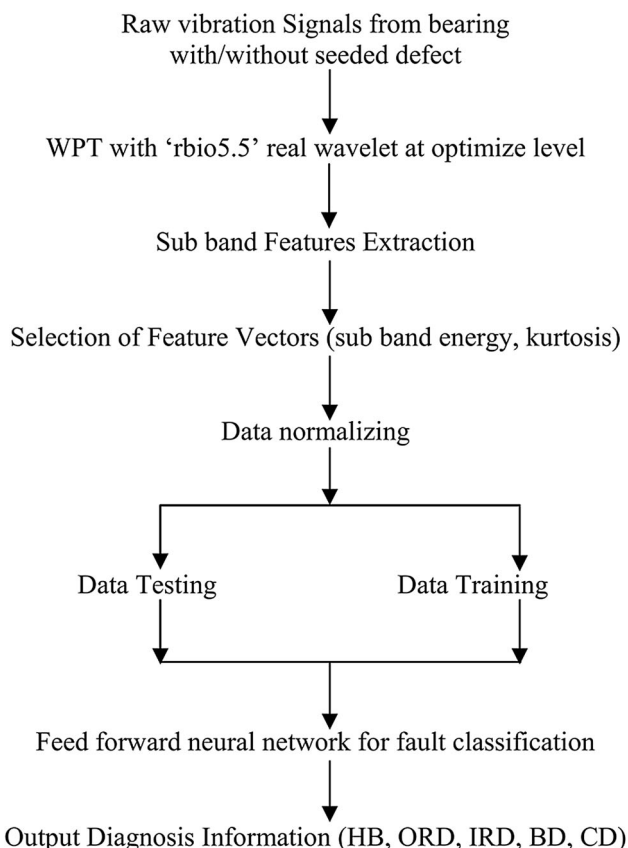


Fig. 8 Flowchart of implementation by Pandya et al. [1]

Normalization of the inputs is then done for the entire samples of 1000 data sets prepared, at a range of 0.1 and 0.9. Each input was equally assigned one of 5 different output classes (4 faulty conditions and 1 normal) as targets. The overall classification rate is highly successful at a rate of 97%.

Feng and Xu [70] used a Kohonen self-organizing NN to investigate the classification of 5 different fault types in liquid propellant rocket engine ground-testing beds. The faults classified by the NN are 5 different fault modes. The research was done using two types of ground-testing bed fault data; a mechanism-based model, and an experts system-based statistical model. While the sample size generated for each model is different, each sample has 5 parameters as input features. The 5 parameters are: pressure of hydrogen reducing valve, pressure of hydrogen tank, pressure of hydrogen pipeline, pressure of hydrogen pump, and flow of hydrogen pipeline. The research is conducted primarily to see the difference of NN performance with and without PCA preprocessing done to the input set, for both model cases, leading to a total of four separate NN tests conducted. Details of the reduced dimension used after PCA were not described. The NN for the mechanism-based model data set was assigned 5 clusters, 4 of which are different fault modes, and one of normal conditions. A total of 150 samples were generated from a simulation model, 100 being used as training and 50 for testing. The NN for the statistical model used 16 total clusters; 15 different fault modes and one of normal conditions. For this mode, 100 samples were generated from the model, 20 being used as training and 80 for testing. For both models tested, the data set preprocessed by PCA performed considerably better at classifying the inputs and took significantly less network training time (from $\sim 92\%$ identification ratio to $\sim 96\%$, and decreased training time by close to 90%).

Taqvi et al. [10] used an MLP LMBP network for detecting anomalous behavior within the operation of a distillation column through classification of faults using steady state output variables as the input vector, which were generated from an Aspen Plus[®] simulation of a distillation column. Normalization was done on the 6 variables in the input vector, and the deviation ranges tested are of $\pm 2.5\%$, $\pm 5\%$, and $\pm 7.5\%$ from nominal operation values. Each sample was classed by a fault as the target, and 6 total potential fault conditions were used. The fault conditions are increment/decrement in feed rate, increment/decrement in reflux flow rate, and high/low steam flow rate. Additionally, the faults were classified based on fault intensities, i.e., the classes were not simply binary values of 0 or 1, but rather a decimal value, between 0 and 1 based on how severe the output variables of the column was. The results show that 5 faults were satisfactorily

diagnosed, and only Fault 1 (decrement in feed flow rate) was not satisfactorily predicted.

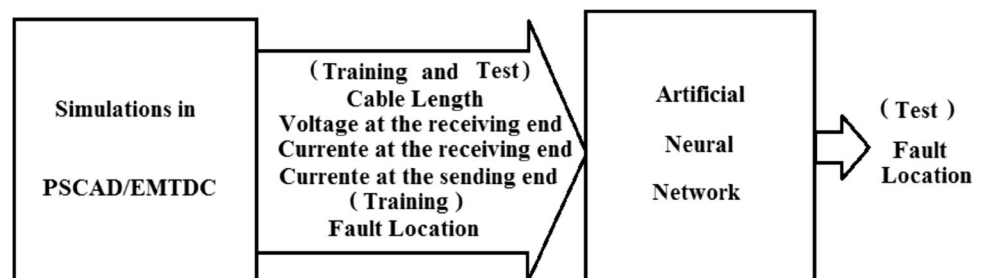
Gastaldello et al. [3] used an MLP (LMBP) network for identifying the location of faults in underground electrical cables through noninvasive means. Two hidden layers were used, and the inputs (of 4 cable variables per sample) and targets (location of fault) are generated from simulation software for use in training as well as testing (Fig. 9). It was noted that the network performance for detecting faults occurring in 300-m-length cables was not successful, likely due to scarcity of samples for said cable lengths. However, looking into its predictive capabilities for various other cable lengths from as low as 100 m up to 4300 m, the average error of fault location predictions was only off by 3.08% when using 326 training samples, and 0.25% when 458 samples were used, which indicates satisfactory performance for every other cable length. The authors noted that further investigation of using other network topologies will be necessary.

Ziani et al. [71] used an MLP for the fault diagnosis of bearings, using GA as the primary optimization algorithm for selection of signal features for use in training. Compared to a similar study [4], the statistical features used are significantly less, and involve fewer NN testing configurations. What is uniquely done is the use of power spectral density (PSD) calculations to obtain spectral features, and a wider range of frequencies sampled for statistical features. Five total signal classes were used, 4 of which is of faulty behavior, and 1 normal. The signals of each class were taken from four different speeds, totaling to 20 unique time-domain samples. The samples were split further into 2 groups, forming 40 sets. The samples are then processed into 4 different frequency bands. From the frequency bands, 6 statistical features which are: root mean square, variance, skewness, kurtosis, normalized 5th moments, and 6th moments, were taken. The statistical feature count is up to 24. Four additional features were sampled by taking the PSD from each of the frequency ranges, totaling the statistical feature count to 28. The global data set consists of 40 time-domain samples (20 for training, 20 for testing) with 28 combined features (24 statistical, 4 spectral). The network performance using this data set has shown to be acceptable with the spectral features, but not so well with

the statistical and combined features. This is likely due to the curse of dimensionality taking effect, as 24 and 28 is much too high of a parameter count for such a small sample size. Additionally, the number of iteration stopped at 7 and 11 for the statistical and combined features, respectively, indicating a problem of convergence toward local minima. When training using features selected through GA (the number of features selected was reduced to 4 and 2), classification performance was at 100% for four random trials.

Sharma et al. [66] used a MLP BP network with momentum for detecting the presence and severity of process faults in an ammonia (NH_3) stripping column. Six classes of faults (low/high feed rate, low/high feed composition, and low/high reboiler vapor rate) were used as targets for the 6 parameters (mol fraction of NH_3 in bottoms, mol fraction of NH_3 in distillate, bottoms temperature, bottoms distillate, bottoms rate, and pressure drop) input vector. The input data set is also varied in a way such that not only does it has a specific fault target, but there are also specific severities of each fault target. The inputs are also normalized, and the discretization of the fault severities was made between the range of [0, 1] for input deviations of $\pm 5\%$, $\pm 12.5\%$, and $\pm 20\%$ from nominal value. This means the target values are equivalent to 0.25, 0.625, or 1.0 for the three different severities, respectively. The objective of the classification is similar to Taqvi's case [10], in that not only the type of fault is to be detected, but also the severity of the fault. It is worth noting that the authors also studied samples with multiple fault classes assigned as targets (double faults and triple faults). The classification efficiency of the outputs was evaluated by the root MSE (RMSE), maximum average percentage error (MAPE), and Pearson correlation coefficient. The terms "recall" and "generalization" used by the authors refer to the training and testing step of the ANN, respectively. The network was reported to be able to detect and classify most of the faults correctly in both training and testing stages except for samples classed with the low feed composition fault. While unmentioned, the performance of the network in the testing stage for severity prediction was not shown to be very consistent for $\pm 5\%$ severity faults, and all levels of severity for the low vapor rate fault. Through the

Fig. 9 Diagram of overall methodology used by Gastaldello et al. [3]



analysis of importance of each parameter using Garson’s method [57], the bottoms composition, bottoms temperature, and distillate temperature were found to be of little significance in contributing to the fault classification performance. A new network using the same samples but only 3 input parameters (eliminating the three parameters of lesser significance) was trained and was found to perform similarly to the 6 input NN for single fault classification (i.e., only low feed composition fault was not classified properly). The multiple fault detection performance was shown to be better using the 3 input NN, compared to the 6 input NN, while triple fault detection was not achievable for both networks.

A similar study was done by Behbahani et al. [72] for a CO₂ absorption/stripping column, using an MLP BP with momentum, using 5 normalized inputs (sweet gas flow, sweet gas temperature, rich solvent flow, rich solvent temperature, and column pressure drop) and 8 classes of faults (low/high flow of feed, low/high concentration of CO₂ in feed, low/high flow of liquid absorber, and low/high concentration of monoethanolamine in absorber). The ranges of fault severity within the samples are of ± 5%, ± 10%, ± 25%, ± 50% deviation from normal values. The classification efficiency of the outputs was evaluated by the MSE and MAPE. The training and testing performance for the prediction of single faults was found to be very successful, along with the detection of test samples containing double faults. Detection of triple faults was not successful, noted by the authors to be due to the training of the network done using only single fault samples.

In another study with a much simpler basis, an attempt at detecting faulty behavior using an RBF network was done for a distillation column containing a binary mixture of toluene/methylcyclohexane by Manssouri et al. [73] Attached to the first output class of “normal operating behavior” (with value of 1) are 6 input parameters which are heating power, preheating power, distillate rate, feed rate, pressure drop, and preheating temperature. An alternative set of data where the heating power is increased by 100% of its stable operating range (leading to changes in preheating temperature and pressure drop) was also prepared, which was labeled as the second class of “abnormal operating behavior” (with value of 0). After training the network with an unspecified amount of data and testing with 50 sets of normal operating data, it was seen that the model managed to classify the operating data as “normal” at close to 80% of its intended output value of 1. When tested with 24 samples of the abnormal operating data, the trajectory of the output class did begin to lean toward the value of 0, averaging around 0.38 across all 24 samples. This is likely due to the relatively small abnormality change of only 3 variables in total, leaving still about 3 of the other variables in near stable operating state.

Mekki et al. [6] used an MLP to model and detect faulty behavior of partial shading of photovoltaic modules. Pre-processed by normalization, the 2 inputs used are temperature and solar irradiance, and the 2 outputs to be predicted are output current and voltage. The network was trained using a multitude of BP algorithms, each at 10,000 epochs, to find the network that gave the best performance regardless of convergence speed. The chosen training algorithm that gave the best performance was resilient BP, which gave an MSE value of 0.00101 and a regression coefficient R^2 of 0.999. The “fault” in this case is the number of shaded photovoltaic cells. The mean absolute error (MAE) between the measured performance (current or voltage) and the estimated performance is used to diagnose the magnitude of this fault. It should be noted that the behavior of the ANN model very closely follows the actual measurement in a linear manner, meaning the R^2 value is close to 1, and the MSE is almost zero at normal operation state, indicating a near flawless model that can be used as a reference. With reference to Fig. 10, a faulty state is detected when the error between the measured and predicted performance exceeds the threshold value S_i and S_v , which are defined as the minimum detectable value of the residual error (number of shaded cells (NSC) required to cause noticeable drop in performance). When a fault is detected (significant shading occurs), the MAE value is

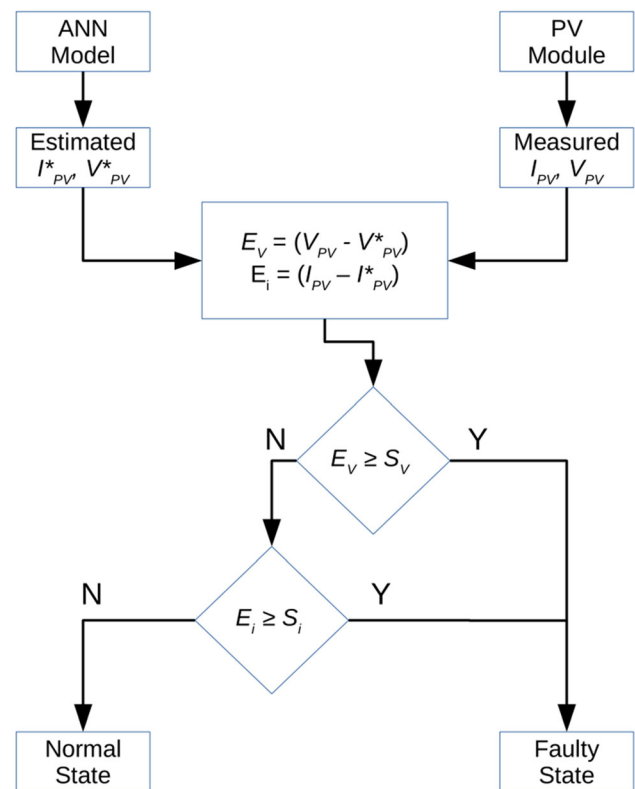


Fig. 10 Fault detection scheme by Mekki et al. [6]

used to diagnose the magnitude of the fault. The developed model manages to obtain a linear behavior based on the MAE and NSC, and the authors also managed to successfully plot the faulty trend (performance loss against NSC) in a linear fashion as well.

Jamil et al. [74] used an MLP (LMBP) for fault detection and classification in an electrical power transmission system. Two separate networks were made for the attempt in detection and classification of a fault. It was noted the inputs from each trial were normalized. The first trial was done using 6 inputs (three phases of voltage and current waveforms) to simply detect whether a fault occurred (Output was set to a binary decision of [0, 1].) The results showed an overwhelming ability to successfully detect the occurrence of a fault. The second trial of fault classification used the same inputs and was assigned 10 target classes represented by 4-dimensional outputs based on one of the three phases of the transmission line and output for the ground. Ten distinct faulty class values were able to be made using in such a manner, e.g., Fault 1 = [1 0 0 1], Fault 2 = [0 1 0 1], etc. The regression fit showed an overall R^2 value of 0.93788, and a classification efficiency of 78.1% from the classification confusion matrix, which was deemed by the authors to be successful in classifying for all 10 faults. A noticeable aspect of the authors' implementation was in the topology of the neural network, totaling five layers with a size configuration of 6–10–5–3–1 (with each number referring to the size of the layer) for the first trial, and 6–38–4 size configuration for the second trial.

With reference to the use of dynamic NNs for dynamic data sets, a few examples of recurrent NN usage are also noted. For the detection and isolation of faults of an industrial gas turbine, Nozari et al. [75] devised an MLP with tapped delay lines for identification of faults, assisted by a local linear neuro-fuzzy model to ensure its fault detection thresholds are robust and adaptable to noise. While the case above [75] utilizes process residuals as the input to the fault predictors, a similar method was performed by Taqvi et al. [76], however utilizing the direct process variables as inputs, for the detection faults in a distillation column using a nonlinear autoregressive networks with exogenous (NARX) NN. Lastly, Kiakojoori and Khorasani [77] used two dynamic NNs; the Elman NN and NARX NN, to predict the simultaneous occurrence of compressor fouling and turbine erosion within a gas turbine engine. For this case, the Elman NN (where the feedback loop occurs within a separate bank of delays in the weight space) reported better performance compared to the NARX NN (where the feedback loop occurs from the output space directly to the weights).

4.2 Standard practices in ANN application

As seen in column 3 of Table 2, a majority of the cases studied (nearly 90%) uses MLP as the preferred NN architecture in their research, utilizing at most 1 or 2 layers. This is mostly due to the higher prominence of supervised learning problems encountered within FID, leaving unsupervised learning as an unpopular option. The preprocessing of inputs is shown to be extremely important in obtaining successful results, as a few of the studies even have included a comparative analysis to show its efficacy over largely unprocessed inputs [2, 4, 20, 67, 70, 71]. Therefore, in the discussion of neural network usage, a more prominent topic to touch upon is more so the method of preprocessing the input space than the neural network usage itself (putting aside the complexity of the applied FID case as a separate issue). The application of the inputs themselves into MLP ANNs is however relatively easy, which enables it as an attractive option for quick testing an input–output behavior model.

A glance of column 4 in Table 2 shows that the use of normalization is practically a standard, and is encouraged to be used at all times. Even in cases where not explicitly mentioned [3, 7, 68], it is highly likely that the analysis methods reviewed had a normalization step incorporated by default. Research involving signals as the source of interest can benefit from wavelet decomposition to extract more salient features to use as inputs in the neural network [1, 20, 67]. For cases where a large amount of parameters are involved per sample, the use of PCA can be of interest to reduce the dimensional properties of the input space to reduce any possible redundancy within the samples, which can promote better network generalization and computational efficiency [2, 5, 20, 70]. This is especially seen from Zhou et al.'s [2] case, where the use of PCA manages to highlight 2 distinct groups correlating to separate behaviors of “faulty” and “non-faulty” ranges from the staggering number of 22 variables. From cases where optimization methods for parameter selection are used, it is seen that a smaller number of parameters in the input space can be completely sufficient to represent the behavior of certain systems [4, 66, 71]. Thus, a conservative estimate of the appropriate number of input parameters for modeling a system using NN can be approximated as between 1 and 6, though it highly depends on the system in mind. Using more than that is certainly an explorable option, though it might come at the cost of higher computational costs and notably higher sample size requirement. If the information is readily available and plentiful in size, it is advisable to proceed with using as many parameters as available possible at first. This is because one solution to overcome the problem of bad NN generalization is to attempt to model

Table 2 Summary of reviewed applications

Authors	Objective	NN architecture	Preprocessing	Source
Hwang et al. (1993)	Detect and diagnose faults on a mathematical model of a pressurized water reactor type nuclear plant	MLP (BP)		[7]
Paya et al. (1997)	Detect and classify 6 different vibrational signals from a gearbox	MLP (BP)	Wavelet (D4), normalize	[67]
Aminian and Aminian (2000)	Determination of faulty components within an analog electronic circuit by impulse response analysis	MLP (BP)	Wavelet (Haar), PCA, normalize	[20]
Kumar (2003)	Detection of various textile defects through segmentation of static images	MLP (LMBP), linear feed-forward ANN	PCA, normalize	[5]
Sharma et al. (2004)	Fault detection and severity diagnosis of an ammonia stripping column	MLP (BP + momentum)	Normalize	[66]
Samanta et al. (2004)	Classification of various faulty bearing behaviors	MLP (LMBP), RBF, PNN	Various statistical feature extraction, genetic algorithms, normalize	[4]
Manssouri et al. (2008)	Fault detection in a distillation column	RBF	Normalize	[73]
Behbahani et al. (2009)	Fault detection of a CO ₂ absorption/stripping column	MLP (BP + momentum)	Normalize	[72]
Feng and Xu (2011)	Performance analysis of fault classification in liquid propellant rocket engine ground-testing beds	SOM (Kohonen)	Normalize, PCA	[70]
Gastaldello et al. (2012)	Prediction of fault location within underground electrical cables	MLP (LMBP)		[3]
Pandya et al. (2012)	Diagnosis of ball bearing fault type based on input features obtained from preprocessed vibration signals	MLP	Wavelet (rbio5.5), sub-band energy + kurtosis feature extraction, PSD, normalize	[1]
Ziani et al. (2012)	Diagnosis of ball bearing fault type based on input features obtained from preprocessed vibration signals, optimized through GA	MLP	Various statistical feature extraction, genetic algorithms, normalize, PSD	[71]
Zhou et al. (2014)	Identification of fault across multiple processes (example case shown for faults of pulmonary readings through parameter behavior reconstruction)	MLP (BP)	PCA, <i>k</i> -means clustering	[2]
Jamil et al. (2015)	Fault detection and classification in an electrical power transmission system	MLP (LMBP)	Normalize	[74]
Mekki et al. (2016)	Modeling and detecting faulty behavior of partial shading of photovoltaic modules	Various MLP BP methods (preferred method chosen as resilient BP)	Normalize	[6]
Taqvi et al. (2017)	Identification and diagnosis of faulty process behavior of a distillation column	MLP (LMBP)	Normalize	[10]
Banjanovic-Mehmedovic et al. (2017)	Detection of anomalous thermal power plant process behavior through classification.	MLP (LMBP), Elman, PNN		[68]
Nozari et al. (2012)	Detection and isolation of faults in an industrial gas turbine, with fault thresholds generated from neuro-fuzzy method	Recurrent MLP (LMBP), Local linear neuro-fuzzy modeling	Normalize	[75]
Kiakojoori and Khorasani (2016)	Prediction, monitoring, and prognosis of engine turbine degradation under multiple fault severities and conditions	NARX (LMBP), Elman recurrent NN		[77]
Taqvi et al. (2018)	Detection of faults in distillation column, with thresholds obtained using Shewhart control charts	NARX (LMBP)	Normalize	[76]

the system behavior with as many features as possible, regardless of its importance. The evaluation of which parameter to discard can be done in the preprocessing step

[4, 66], or if either parameter numbers are small enough or processing time will not be an issue, they can be discarded manually through simple trial and error. Conversely, if the

network is seen to be lacking robustness, the addition of more sample sizes with wider data variation ranges can be a suggestible action to take [3].

4.3 Characteristics of the reviewed cases

The main characteristics selected for evaluation from each case are shown according to Table 3, which is much more specific than the general summaries of the cases shown in Table 2. Within Table 3, the characteristics are organized according as; type of problem the authors are concerned with in the ANN application (2nd column), input parameters of the network (3rd column), classification format of the data seen in the targets and output (4th column), and whether a static (with sample size shown) or dynamic data set is involved (5th column). They are chosen based on the input–output sizes used in constructing the model and required analysis within each case. Unless annotated otherwise, the selected inputs and outputs are from the models with the best performance in tandem with the most efficient preprocessing configuration (for the case where multiple ANN architectures are tested). It is done in this manner to be made appropriate for the intended audience of this paper; new researchers wanting to explore ANN as an option for performing data-driven analysis by looking into how it was done by others.

What is key in ANN usage is the proper establishment of both the materials with which the black-box model has to work with (the input) and how both the researcher and machine can interpret the result (the output). A thorough discussion of ANN's performance relative to other process history-based FID methods is explored by Venkatasubramaniam et al. [47], with a heavy emphasis on chemical process application. The discussion was made particularly by comparing the capabilities of ANN against other options for process control that is available at the time. The comparison was done using semiquantitative characteristics such as quickness of diagnosis, isolability, robustness, among others. In comparison with the classical options of using an observer and expert systems, ANNs are highly regarded for its quick detection speed, good isolation capabilities, robust generality through learning, novelty detection capabilities, and compact storage options. Their main drawback is the lack of explanatory detail behind the model development and function (due to its black-box nature). In comparison with another data-driven model called Bayesian networks (BN) done by Tidriri et al. [78], both are shown to be rather equal in capabilities in similar measures of robustness, calculation speed, etc. However, the BN method suffers from the requirement of various prior probability data, a problem owing to the large training data required of the system beforehand. While ANN is described largely in a positive note by the authors, many of

the disadvantages associated with ANN in the review of Venkatasubramaniam et al. (lack of multiclassing capabilities, inability to describe classification errors) are mostly alleviated today by the increasing development in not only the algorithms used within the ANN models, but also the accessibility to such new technologies for the average end user, partly in thanks to the inclusion and simplifications done by numerical computation software developers of products such as MATLAB and TensorFlow.

Furthermore, the vast difference between the types of problems encountered by each of the studied cases cannot be assessed by general descriptors alone. An example of this is in the analysis of robustness and adaptability. Within any arbitrarily selected case, its robustness (or flexibility) can only be judged based on how well designed the inputs are made to cover the expected behavior within the scope of the studied phenomenon. For example, multiple class problems within electrical transmission lines are far too different than multiple faults within a chemical process to be reliably compared. Whether an ANN is capable of generalizing the studied behavior is highly reliant on the researcher's expertise and interpretation of the specific subject matter, and how it is expressed through the input and output design. In a similar vein, it can be said that neural networks both lack adaptability in the sense that it always requires the reconstruction of a new set of data for each new system required to be studied, while at the same time is adaptable by definition of its design as a universal function approximator. This may not always be the case depending on how the contextually broad the problem is defined, though such cases are mostly an exception rather than the norm. Even within the similarly established ball bearing fault cases [1, 4, 67], disregarding the different type of experimental methods used for obtaining data, there are unstated physical differences such as type of motors used to generate the vibrational signals, quality of ball bearings, manufacturer brand, and so on.

4.4 Problem type assigned to each case

In this minor subsection, the characteristic of problem type from each reviewed cases (column 2 of Table 3) are examined. The labels for each problem type are made based on several criteria. If the faults are known to some extent by the experimenter, and a simple qualitative label was to be required for each characteristic fault based on the behavior of the inputs, the type of problem is labeled as "fault type classification." The salient features from these problems are the deterministic classes, i.e., a specific numerical value is assigned for each class. Outside of any possible difficulties with the input parameter selection, it is one of the easiest problems to solve using ANN in terms of implementation and analysis. Any form of increasing

Table 3 Summary of characteristics for each reviewed application

Authors	Problem type	Inputs	Outputs (and targets, for supervised learning)	Dynamic or static	Source
Hwang et al. (1993)	Parameter identification, error threshold generation, and fault detection	4 reactor variables + time, 2 variables (time + coefficient), and 3 past steam pressure residuals	Coefficient for predicting process behavior, 8 error thresholds, and type of fault	Dynamic	[7]
Paya et al. (1997)	Fault type classification	Dominant wavelet number and amplitude (per fault class)	6 different classes (5 faulty, 1 normal)	Static (30 samples for each of 6 class, totaling 180)	[67]
Aminian and Aminian (2000)	Fault type classification	Wavelet coefficients from impulse response signal (reduced by PCA, taken from 5 different levels of wavelet decomposition)	9 different faulty class values (higher or lower values from nominal readings)	Static (50 samples for each of 9 classes, totaling 450)	[20]
Kumar (2003a)	Fault detection	Image vector matrix reduced by PCA (dependent on type of defect)	Characteristic image of segmented defects	Static	[5]
Kumar (2003b)	Linear classification	Color normalized projection line (horizontal and vertical line, on 2 separate networks)	Decision of either “defect” or “non-defect” on that line (through error calculation between line’s color and reference color)	Static (640 vertical signals, 480 horizontal signal)	[5]
Sharma et al. (2004)	Fault detection and diagnosis	3 process variables (cut down from 6 with Garson’s equation of relative variable importance)	6 types of fault at severities between $\pm 5\%$ $\pm 20\%$, inclusive of double faults and triple faults (output containing multiple classes)	Static	[66]
Samanta et al. (2004)	Fault type classification	Various parameters generated from signals of vibrating ball bearings (6 parameters used in most optimized network)	2 classes (faulty vs non-faulty); 2 vector output for MLP and RBF, 1 vector (1 or 0 value) for PNN	Static (288 samples, each with 45 features before preprocessing)	[4]
Manssouri et al. (2008)	Fault type classification	6 process variables	2 class (normal operation, abnormal operation)	Static (74 samples for testing; of which 50 is normal, 24 is abnormal)	[73]
Behbahani et al. (2009)	Fault detection and diagnosis	5 process variables	8 types of fault at severities of $\pm 5\%$, $\pm 10\%$, $\pm 25\%$, and $\pm 50\%$ (separate testing of double faults are successful, failed for triple faults)	Static (200 for training, 60 for testing)	[72]
Feng and Xu (2011)	Fault identification	5 parameters (150 samples each; 100 for training, 50 for testing)	Self-generated groups/clusters of faults	Static	[70]
Gastaldello et al. (2012)	Fault detection and diagnosis (through curve fitting)	4 parameters relating to the cable properties	Location of cable fault	Static (458 samples)	[3]
Pandya et al. (2012)	Fault type classification	12 parameters (3 sub-band energies + 3 kurtoses, each having horizontal and vertical parts) extracted from wavelet transform	5 different classes of faults	Static (1000 samples, 70% train, 20% validation, 10% test)	[1]
Ziani et al. (2012)	Fault type classification	2 parameters (selected from 28 with GA)	5 different classes (1 normal, 4 faulty)	Static (20 for training, 20 for testing)	[71]
Zhou et al. (2014)	Time series prediction	22 time variant parameters taken from 10 past time instances	Predicted behavior of 22 parameters at 11th time instance, and onwards	Dynamic	[2]

Table 3 (continued)

Authors	Problem type	Inputs	Outputs (and targets, for supervised learning)	Dynamic or static	Source
Jamil et al. (2015)	Fault type classification	6 inputs (voltage and current readings at 3 different phases)	4 output nodes (of phases A, B, C and ground), totaling to 10 distinct faults inclusive of multiple class faults	Static (792 for each class, totaling 8712)	[74]
Mekki et al. (2016)	Curve fitting/model generation	2 parameters; temperature and solar irradiance	2 outputs; current and voltage generation	Static (3000 samples, 70% training, 30% testing)	[6]
Taqvi et al. (2017)	Fault type classification	6 process variables	6 types of faults at severities of $\pm 2.5\%$, $\pm 5\%$, and $\pm 7.5\%$	Static	[10]
Banjanovic-Mehmedovic et al. (2017)	Fault type classification	6 process variables	2 classes of either “normal data” or “anomaly data”	Static	[68]
Nozari et al. (2012)	Fault detection and isolation	2 input variables; valve angle, fuel flow	4 output variables; (compressor torque, outlet temperature, combustion chamber outlet temperature, pressure)	Dynamic	[75]
Kiakojoori and Khorasani (2016)	Fault detection and diagnosis	1 process variable; fuel flowrate	1 process variable; turbine output temperature	Dynamic	[77]
Taqvi et al. (2018)	Fault detection	3 input variables; (top and bottom compositions, column pressure)	3 output variables; (reflux flow, reboiler duty, condenser duty)	Dynamic	[76]

complexity will mostly be from the preprocessing done to obtain the features and the initial problem formulation of the class group types, which will vary in difficulty based on how intricate the studied system is. On the other hand, if there are no observable distinct groups, unsupervised learning through a self-organizing ANN is another alternative tool that will assist in both affirming or rejecting any hypothetical fault classes that may or may not be present. If the number of classes to be considered is seemingly higher than expected, Jamil et al.’s [74] case can be used as an example to follow. While a higher class count might necessitate a larger output vector, it might also require an exponentially larger training sample size. To overcome this problem, Jamil et al. used a combination of permutative vector space to map out all fault classes and a higher than usual hidden layer size of 36 to expedite the training process. However, seeing as the reported prediction performance in the testing stage is imperfect on its own right, it is still recommended to use the n -ary style of class formatting to cover all faults (as shown in Sect. 3.3) for the initial attempt.

In cases where the outputs carry both qualitative and quantitative properties, the problem is labeled as “fault detection and diagnosis.” It is labeled as such mainly due to how the faults have not only a specific class, but also a level of how severe the fault can grow depending on the

possible abnormality range of the input parameters. In a sense, it is very much similar to a curve fitting problem with multiple dependent variable terms; however, the term values are segregated between constraints. Mostly represented by the chemical process fields in this review [10, 66, 72], the level of severities is typically there to allow for a fault class to be expressed in varying degrees of harm levels such as “tolerable” or “critical.” If an accurate model of faulty process behavior can be achieved using the process variables alone as indicators, it may serve as an alternate fallback simulation to find out if a fault can be detected proactively before it may even occur. Outside the purpose of FID, it can also be used to observe how much quantitative influence a particular variable has toward one type of output variable in a more realistic and non-isolated condition.

As stated in Sect. 2, loss functions and the performance metrics are not necessarily the same. The use of MSE (mean square error) is widely popular and is regarded as the most commonly used loss function [79] in most ANN architectures. Despite this shared similarity, it was seen that the performance criteria are never explicitly consistent across all the cases studied. In the classification cases for example, most of the networks use MSE as the loss function; however, the manner in which the results were discussed was done using concepts of classification accuracy

[1, 4, 20, 67, 74]. Depending on the problem, the metrics used in analyzing the performance of the trained network will differ. For a standard fitting or modeling task where outputs are evaluated quantitatively (such as the cases of Gastaldello et al. [3] and Mekki et al. [6]), the combined use of MSE and R^2 is usually sufficient in determining the accuracy of the network in modeling the expected behavior between the inputs and outputs. For cases where a deterministic fault class is preferred (such as the qualitative labeling of Fault 1 as [1, 0], Fault 2 as [0, 1], etc.) the use of true/false classification is indispensable for the discussion of network performance. The many statistical classification derivatives such as true positive rate (recall/sensitivity), true negative rate (specificity), confusion matrix of overall classification performance, and many others would be the additional suggested metric to use, based on the analytical needs of the study. An example being in process control loop monitoring, where a system producing false positives can be extremely dangerous, necessitating a higher priority for models that perform well at obtaining true positive performance [80].

Another unexplored advancement among all the studied cases, however, is the use of cross-entropy (CE) as a performance metric. Unlike MSE (as shown in (4) where y is the actual probability, and \hat{y} is the predicted probability over number of samples N), which processes each sample set as with a functional real-value output, CE looks into the probability of one set of data belonging to either one of the 2 classes within the target vector. Compared to the absolute type of error generated by the MSE, the error type of CE is relative to the number of classes used in training, allowing outputs to be expressed through more than just real values of [1, 0] or [0, 1]. Depending on how correlated a particular sample set is to a certain class, CE will also take into account the weight of a predicted probability such as [0.76, 0.24] or [0.19, 0.81] relative to the total class targets trained. Note that this is mainly assuming the softmax activation function is used, so the classes are normalized between 1 and 0. While this does seem inapplicable to fault detection and diagnoses cases with varying severities, it is still suggested for use in cases where the classes are deterministic and dependent on each other. Even when used as a loss function, it has the ability to more accurately calculate posterior probabilities, especially within a limited amount of samples, when compared to MSE [81]. While it has also shown promise for use as a loss function in training classification problems, its use as a performance metric is noteworthy on its own. Shown in (5) is the simplified version of the CE index performance metric for a two-class application, where H is the average cross-entropy score. A lower overall score is more desirable, and it can easily be applied over the outputs generated from a network's training or testing session, as another means of

numerically assessing the performance of a classification network with deterministic classes.

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (4)$$

$$H(y, \hat{y}) = \frac{1}{N} \left(- \sum_{i=1}^N y_i \log(\hat{y}_i) \right) \quad (5)$$

4.5 Input and output format within the ANN of each case

In this subsection, the characteristic of inputs and outputs used in each reviewed cases (columns 3 and 4 of Table 3) is examined. This is the stage where the context of the research matters most, and the best course of action is to use due diligence and design the most appropriate input vector data and the outputs needed from them using whatever data that are most appropriate for the FID problem in mind. If signals or any image-like data are concerned, feature extraction through the use of statistical moments (especially when physically generated signals are involved [4, 71]), PCA, and wavelet transform is highly recommended. In fact, the methods described in Sect. 3 are only a minute fraction of the many data transformation strategies employable to obtain a varying data set format. The use of a large sample size just as important to ensure that there is sufficient information to ensure that the behavior learned of the system is robust to generalization. It is also important to remember that the behavior learned from the ANN is only as good as the inputs and outputs fed into it.

A rather difficult task highlighted across the cases is the prediction of parameters with multiple class targets. While it is successful for the prediction of the occurrence of up to 2 simultaneous faults in all the cases, the results for 3 simultaneous faults leave a lot to be desired. Even in Jamil et al.'s [74] case where the prediction was considered a success, the confusion matrix classification percent leaves a lot of room for improvement. The use of SVM, Bayesian network, and other machine learning techniques that are also suited for classification tasks [78] can be considered as an alternative if multi-class target problems are needed to be solved.

On a separate tangent, it was noticed that there was a lack of clarity in presentation of inputs and outputs in some of the papers studied. While a few of them can be excused as the earliest entries in ANN use within FID, the lack of clarity in data presentation observed (no numerical tables showing the input parameter set, no clear distinction between total sample size per class and input vector size, odd clumping up of both training and testing data set) can

be a careful reminder of how important it is to ensure the input and output data format is arranged and presented as clearly as possible, either through graphical or tabular means. As this issue is mostly pervasive in older papers, it is likely that the increasing availability of accessible ANN tools and resources over time resulted in a natural progression toward higher understanding among ANN users in documenting their methodologies. Hwang et al.'s [7] paper stood out as rather interpretable, despite being the oldest out of the studied cases. The difficulty in understanding applications of ANN, particularly of the cross-disciplinary variety, mostly stems from the fact that it is a foreign subject matter to mostly all but the authors and the interest groups surrounding the case or field in which it is applied. In order to relieve this problem to some extent, researchers are encouraged to explore ways for further reducing the premise of the ANN input–output structure in a clearer manner (wherever and whenever appropriate, of course) so that it may be better appreciated, and perhaps even be adapted, by audiences from outside the particular expertise in which the ANN is applied. The easiest course of action is to include a full diagram of the ANN structure, describing clearly the number of inputs entering the ANN (parameters and targets) as well as the outputs. If the inputs themselves are obtained through relatively complicated preprocessing steps, it would be extremely helpful to include a flowchart describing the data transformation in a step-by-step manner from the starting source material into the final format that will be used as inputs or targets. Additionally, a sample table showing a brief but exact set of the numerical values of the input or output, e.g., as seen in Pandya et al.'s [1] and Gastaldello et al.'s [3], would also greatly assist in the direct numerical demonstration of what goes in and out of the ANN for easier reference to the reader. An exemplary format that highly improves readability of the experiment and results can be seen from Banjanovic-Mehmedovic et al.'s [68] case. The authors have condensed the information of the input data and the NN performance through a graph of how the inputs vary in a minimalistic manner by previewing 3 parameters across a time-sample axis, and have done the same to clearly show the classification performance of varying network types. In the end, while this may have no bearing on the experimental results whatsoever, clearer documentation is always crucial for posterity.

4.6 Static (time invariant) or dynamic (time variant) data set

In this subsection, the characteristic of problem type from each reviewed cases (column 5 of Table 3) is examined. The main difference to be understood between static (discrete) or dynamic (varying with time) data is whether the

output is influenced by the parameters alone, or if it is also influenced by how the input changes with time. An easy example to consider is the ball bearing application examples [1, 4, 67]. The vibrational inputs are associated as having a direct relation with the type of fault in the form of a discrete time data set. For FID applications in which live monitoring is a concern, the use of dynamic time is much more prominent. The main tasks requiring dynamic time data are for step-ahead/forward predictions, or for modeling time-sensitive data for faults involving an iterative loop of some sort (for which the latter case, recurrent neural networks are a viable option to consider). In cases where static data sets are used, the total sample set of data used for the training and testing of the network are shown (in cases where explicitly stated by authors). This is merely to show the range of sample data counts involved in the learning process of the ANN, for the readers to consider as a gauge. It should be noted that the parameter count is just as important, and the consideration for total number of sample sets to gather should go hand in hand with the number of features to use per data set.

Static NNs alone are supposedly unfit for solving dynamic problems; however, the limitations can be overcome. In the case of Hwang et al. [7], the limitations of dynamic data were resolved by using multiple neural networks as an artificial form of multiple “memory” states. While Zhou et al. [2] posed a problem that was inherently dynamic in nature, the vast number of interrelated parameters played a much larger role in influencing the output, rendering it as a solvable problem using only several time instances backwards as the input (not more than 10 time instances), without the inclusion of any tapped delay lines for the training process. In the three cases where recurrent NNs are explicitly used, the time instances reaches a rather realistically longer length, which necessitates the use of an NN capable of providing dynamic solutions. In Taqvi et al.'s case [76], the investigated state space reaches within the 1–2 h length time span, while the turbine engine problem explored by Kiakojoori and Khorasani [77] requires prediction of reportedly up to 200 cycles of operation under highly intensive conditions. Therefore, for the consideration of problems involving dynamic data sets, it is important to look at the projected time range to be explored for the case before deciding to choose a simple static NN, or begin immediately with a dynamically purposed NN.

5 A brief application tutorial

To describe the relative significance behind each choice to be made when using NN for solving a problem, this section will show the steps taken to tackle an FID problem

previously unexplored with NN; detection of a control valve suffering from stiction (static friction). Valve stiction is defined by Choudhury et al. [82] as the sticky resistance of a valve, represented by both the deadband and stickband (S) when it is opening from a stationary position, and a slip jump (J) representing the sudden release of potential energy stored by the actuators due to static friction in the form of kinetic energy as the valve is moving, in the intermediate process of adjusting the flow rate. This problem is interesting due to the relative difficulty of spotting a valve suffering from stiction without the use of invasive testing methods [83], with various non-invasive stiction detection methods reporting considerably varying results [84]. While the sticky condition is known to have quantifiable levels of severity like most faults, it is still important to be able to first identify its appearance. Attempts to quantify the severity of valve stiction using NN were conducted before by Farenzena and Trierweiler [85]; however, a simple class-based approach was not attempted, thus only rendering the work as useful if the valve was known to be suffering from stiction a priori. Detection of stiction with NN was reportedly done before within the work of Venceslau [86]; however, the case within does not show explicit results demonstrating how the developed NN was able to differentiate between valves suffering from stiction or other possible problems. In essence, the scope of the problem is to solve a pattern recognition problem through classification using NN. Using the Neural Network Toolbox in MATLAB, each subsection that follows is the order in which the problem is handled.

5.1 Input and output formatting (with preprocessing)

Since the analysis will be made with a distinction of either “stiction” or “non-stiction,” the output will be established as a simple binary vector. A large range of oscillatory conditions may be sampled within the non-stiction class; however, for this example the output classes are made considering only 3 other cases: a signal with no oscillation, an oscillatory signal from aggressive tuning, and a signal carrying external oscillatory disturbance. The 3 faulty cases will be simulated separately; however, they all share the target label of “non-stiction.” The case of selecting the right input form goes hand in hand with the preprocessing step. It is noted that the normalization step (from (3)) is considered to be a default step as part of the preprocessing.

For the case of valve stiction, the parameters to be considered are the output from controller signals (OP) and the process variable (PV) response signals. While the actual valve position within the actuator can be a key parameter in identifying a sticky valve, such information is not always

available unless smart valves (which are expensive not always available) are in place. From the previous work, several features were extracted from the PV–OP signals for use [85] as input parameters. For this example, the discrete time data from the PV and OP signals will be directly used as input materials, without any feature extraction. While this can be improved upon later on, it is assumed that 500 time instances of PV and OP will contain enough oscillatory information that can represent stiction behavior. In order to fit the properties of both PV and OP into a single parameter [86] (to allow its use in the inputs), a transformation will take place where the *i*th time iteration (from both PV and OP) is converted into a single parameter, *D* as shown in (6). For our case, the value of *i* is designated as 500. Note that this limitation of requiring exactly 500 time instances will carry on for any future inputs wanting to obtain an output from the final NN developed.

$$D_i = \sqrt{(PV_i - PV_c)^2 + (OP_i - OP_c)^2} \tag{6}$$

where *D_i* is the new value of the particular PV–OP time instance, *PV_c* is the mean of the 500 sampled PV time instances, and similarly *OP_c* is the mean for the 500 OP time instances.

With the input and output premise established, the training data are ready to be generated using a process simulation modified with a stiction model. Using Choudhury’s stiction model, the stickiness of a valve can be varied based on the stick + deadband (S) and slip jump (J) parameter. Through multiple variances of S and J parameters, 164 samples representing stiction behavior were generated, and a combined number of 108 samples were generated representing non-stiction behavior, totaling to an input sample size of 272. The steps taken for generating each individual sample are summarized in Fig. 11. The complete input vector in the form of *D_{i,j}* where *i* is the time instance and *j* is the *j*th data set, is shown in (7). Finally, a table partially showing the values representative of both fault classes, within samples 1–5 (from non-stiction samples) and 268–272 (from stiction samples) of the input, *D* at each time instance *t*, and their respective targets, are shown in Table 4.

$$\begin{bmatrix} D_{1,1} & D_{2,1} & D_{3,1} & \cdots & D_{500,1} \\ D_{1,2} & D_{2,2} & D_{2,3} & \cdots & D_{500,2} \\ D_{1,3} & D_{3,2} & D_{3,3} & & D_{500,3} \\ & \vdots & & \ddots & \vdots \\ D_{1,272} & D_{2,272} & D_{3,272} & \cdots & D_{500,272} \end{bmatrix} \tag{7}$$

5.2 Selection of NN architecture and heuristics

It is advised that scaled conjugate gradient is used as the training method for problems involving pattern

Fig. 11 Block diagram of steps involved in the creation of a single sample

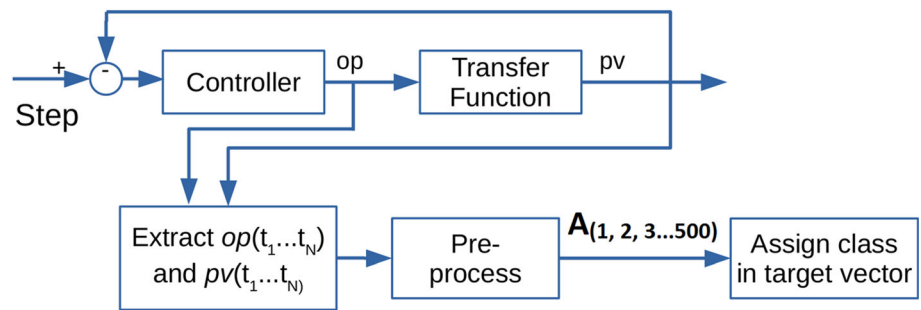


Table 4 Partial sample of input and output space for the valve stiction detection neural network

	D									Output/target vector
	1	2	3	4	5	6	...	500		
Sample 1	1.64932	1.77929	1.96846	1.72238	1.85769	0.38597		0.14392	1	0
Sample 2	0.07311	0.44778	0.21135	0.13658	0.04711	0.29733		0.72089	1	0
Sample 3	0.11268	1.01943	0.47849	0.47019	0.51168	0.23771		0.79119	1	0
Sample 4	0.08839	0.59679	0.36629	0.32501	1.07678	1.06179		0.19313	1	0
Sample 5	0.20078	0.92713	0.60762	0.34494	0.87157	0.41849		0.25313	1	0
:										
Sample 268	2.46043	1.84422	2.01761	1.88621	2.85631	1.97594		2.03201	0	1
Sample 269	2.03263	2.02271	2.03743	1.71096	1.86786	1.88185		1.20506	0	1
Sample 270	1.55363	1.30276	1.29516	1.89016	1.41655	1.07027		1.61999	0	1
Sample 271	1.64008	1.61751	1.69076	1.9358	1.43621	2.05148		2.11358	0	1
Sample 272	2.49685	2.50854	1.82463	1.74064	2.31479	1.95953		1.44422	0	1

recognition, with softmax as the activation function in the output layer [54]. For the hidden layer, a standard tangent sigmoid activation function is used mostly by rule of thumb. Initial training of the NN using neuron sizes between 8 and 12 does not show marginal improvement in performance; therefore, the default value of 10 was used. Two layers in total were employed, comprising one hidden layer and one output layer. The selected structure of the NN used in this example is shown in figure.

5.3 Cross-validation and performance evaluation

The performance of the NN is evaluated using two metrics of cross-entropy (CE) and classification accuracy. What is unique regarding this study is the availability of industrial loops from real process plants which can be used for benchmarking. The industrial loops themselves contain samples where stiction is known to be present, as well as samples that show oscillations due to various other known causes. From Fig. 12 it was seen that the NN was able to learn well due to the steady decrease with no large jumps in the CE value, indicating a smooth progression toward a global solution. The most important indicator to look at in a classification problem is the accuracy. From the cross-validation performance results shown in Fig. 13, the results

are deemed as acceptable for the current application, especially when considering the CE values obtained which are 0.4434 for the training stage, 0.8146 for the validation stage, and 0.8343 for the testing stage. The confusion matrix does show, however, that there is a noticeable bias toward assigning a “stiction” class output (Class 2 in the confusion matrix), rather than “non-stiction.” Further re-training of the network does not yield a lower CE result, showing that it has learned to the best that the training samples will allow. This shows that the 3 oscillatory cases outside of stiction that was lumped together into a single class of “non-stiction” could perhaps be done more elegantly by separating them into their own classes and including more samples. The ratio used for the cross-validation split is at a ratio of 70–15–15. The current model is then exported as a network file, ready to be tested with external data.

5.4 Further performance evaluation

With the availability of benchmarking data, the robustness of the developed network can be further put to the test to see if it has managed to learn enough to generalize across various different conditions where stiction may occur. The loops within the industrial benchmark consist of loops

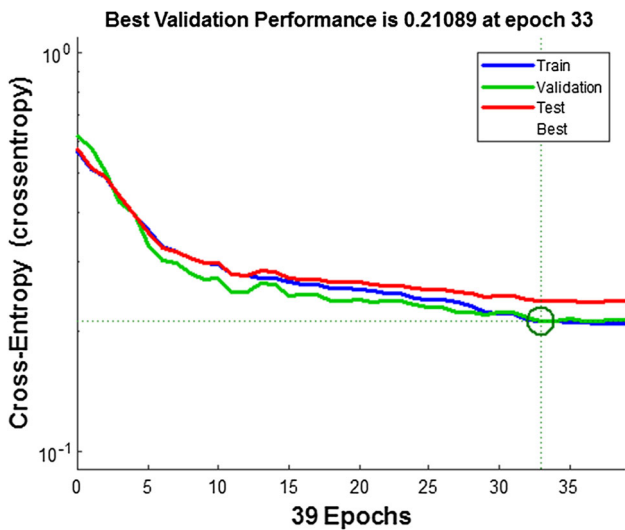


Fig. 12 Learning performance of the NN

across many industries including chemicals, pulp and paper mills, commercial building, and metal processing [84]. From the 93 data sets, we have used 80 to verify the stiction detection model that was developed.

The classification performance of the developed NN against the benchmark test is shown in Fig. 14. Across 43 loops labeled as stiction, 34 were correctly predicted as stiction, while across 37 non-stiction loops, 27 were predicted correctly. While some of the output vectors assigned are likely on the fuzzier side (such as giving results of [0.53, 0.47]), a roughly 75% averaged accuracy for predicting the industrial loops is considered to be very good in comparison with other stiction detection methods [87], which has results varying from 50 to 80% detection rate. Therefore, it can be concluded that even though the input format was rather crude, in the sense that (1) no feature extraction was done, (2) a large discrete time data set was directly used instead to represent each sample, and (3) a rather small training sample size used; the predictive performance was shown to be able to be very robust to various

Fig. 13 Confusion matrix of classification performance from cross-validation stage



types of process loops when tested against real samples from industrial data.

6 Conclusion

Neural networks have seen a lot of interesting applications for a long time across many fields. It functions as a simple black-box model to study the relationship between the variables of a system. In this review, its application in solving various FID problems in engineering is explored. On its own it can be a valuable tool; however, its accuracy in modeling a particular system behavior can be drastically improved by preprocessing the inputs into a more conducive format for the neural network to study. From all the cases looked into, neural networks have definitely shown its worth in solving FID problems for various engineering disciplines. The steps involved in formulating the research premise for studying FID with neural networks are made simple enough due to developments of simpler and more intuitive computational software. With the ever increasing ease of accessibility to the tool, it can now be considered as a standalone or additional testing method for use in corroborating the results of not only FID problems, but many others as well. Using a demonstration, it was seen that even something as simple as directly using discrete time data without advanced preprocessing has shown how easy and effective neural networks are in terms of constructing the premise up to the stage of utilizing the developed network with interpretable results.

Confusion Matrix

	1	2	
1	34 42.5%	10 12.5%	77.3% 22.7%
2	9 11.3%	27 33.8%	75.0% 25.0%
	79.1% 20.9%	73.0% 27.0%	76.3% 23.8%
	1	2	

TargetClass

Fig. 14 Confusion matrix from industrial loop benchmarking

Acknowledgements The authors would like to thank MOSTI Grant Science Fund 0153AB-B67 for the funding provided for this work. The authors would also like to thank Universiti Teknologi PETRONAS (UTP) for the support provided for this research.

References

- Pandya DH, Upadhyay SH, Harsha SP (2012) ANN based fault diagnosis of rolling element bearing using time-frequency domain feature. *Int J Eng Sci Technol* 4(6):2878–2886
- Zhou J et al (2014) Fault detection and identification spanning multiple processes by integrating PCA with neural network. *Appl Soft Comput* 14:4–11
- Gastaldello D et al (2012) Fault location in underground systems using artificial neural networks and PSCAD/EMTDC. In: *IEEE 16th international conference on intelligent engineering systems (INES)* 2012. IEEE, Lisbon, pp 423–427
- Samanta B, Al-Balushi KR, Al-Araimi SA (2004) Bearing fault detection using artificial neural networks and genetic algorithm. *EURASIP J Adv Signal Process* 2004(3):785672
- Kumar A (2003) Neural network based detection of local textile defects. *Pattern Recognit* 36(7):1645–1659
- Mekki H, Mellit A, Salhi H (2016) Artificial neural network-based modelling and fault detection of partial shaded photovoltaic modules. *Simul Model Pract Theory* 67(Supplement C):1–13
- Hwang BC, Saif M, Jamshidi M (1993) Neural based fault detection and identification for a nuclear reactor. *IFAC Proc Vol* 26(2, Part 5):547–550
- Patton RJ, Frank PM, Clark RN (2013) *Issues of fault diagnosis for dynamic systems*. Springer, Berlin
- Rajakarunakaran S et al (2008) Artificial neural network approach for fault detection in rotary system. *Appl Soft Comput* 8(1):740–748
- Taqvi S et al (2017) Artificial neural network for anomalies detection in distillation column. In: *Modeling, design and simulation of systems: 17th Asia simulation conference, AsiaSim 2017, Melaka, Malaysia*. Springer, Singapore
- López-Mata E et al (2016) Development of a direct-solution algorithm for determining the optimal crop planning of farms using deficit irrigation. *Agric Water Manag* 171:173–187
- Choudhury S, Jain M, Shah S (2008) Stiction-definition, modelling, detection and quantification. *J Process Control* 18(3–4):232–243
- Chen J, Patton RJ (1999) *Robust model-based fault diagnosis for dynamic systems*. Kluwer, New York, p 354
- Schmitz GPJ, Aldrich C, Gouws FS (1999) ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Trans Neural Netw* 10(6):1392–1401
- Muhammad T, Halim Z (2016) Employing artificial neural networks for constructing metadata-based model to automatically select an appropriate data visualization technique. *Appl Soft Comput* 49(Supplement C):365–384
- Zhang G, Eddy Patuwo B, Hu MY (1998) Forecasting with artificial neural networks: the state of the art. *Int J Forecast* 14(1):35–62
- Dougherty M (1995) A review of neural networks applied to transport. *Transp Res Part C Emerg Technol* 3(4):247–260
- Jayas DS, Paliwal J, Visen NS (2000) Review paper (AE—automation and emerging technologies): multi-layer neural networks for image analysis of agricultural products. *J Agric Eng Res* 77(2):119–128
- Catelani M, Gori M (1996) On the application of neural networks to fault diagnosis of electronic analog circuit. *Measurement* 17(2):73–80

20. Aminian M, Aminian F (2000) Neural-network based analog-circuit fault diagnosis using wavelet transform as preprocessor. *IEEE Trans Circuits Syst II Analog Digit Signal Process* 47(2):151
21. Spina R, Upadhyaya S (1997) Linear circuit fault diagnosis using neuromorphic analyzers. *IEEE Trans Circuits Syst II Analog Digit Signal Process* 44(3):188–196
22. Dreiseitl S, Ohno-Machado L (2002) Logistic regression and artificial neural network classification models: a methodology review. *J Biomed Inform* 35(5):352–359
23. Patan K (2008) *Artificial neural networks for the modelling and fault diagnosis of technical processes*. Springer, Berlin
24. Hussain M (1999) Review of the applications of neural networks in chemical process control—simulation and online implementation. *Artif Intell Eng* 13(1):55–68
25. Bhat NV et al (1990) Modeling chemical process systems via neural computation. *IEEE Control Syst Mag* 10(3):24–30
26. Miller WT, Werbos PJ, Sutton RS (1995) *Neural networks for control*. MIT Press, Cambridge
27. Antsaklis PJ (1990) Neural networks for control systems. *IEEE Trans Neural Netw* 1(2):242–244
28. Koivo HN (1994) Artificial neural networks in fault diagnosis and control. *Control Eng Pract* 2(1):89–101
29. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. *IEEE Trans Neural Netw* 1(1):4–27
30. Nelles O (2013) *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer, Berlin
31. Rutkowski L, Rutkowski L (2004) *New soft computing techniques for system modeling, pattern classification and image processing*. Springer, Berlin
32. Zhang J, Man K (1998) Time series prediction using RNN in multi-dimension embedding phase space. In: 1998 IEEE international conference on systems, man, and cybernetics. IEEE
33. Haykin S (1994) *Neural networks: a comprehensive foundation*. Prentice Hall PTR, Upper Saddle River
34. Janczak A (2004) Identification of nonlinear systems using neural networks and polynomial models: a block-oriented approach, vol 310. Springer, Berlin
35. Patan K, Parisini T (2005) Identification of neural dynamic models for fault detection and isolation: the case of a real sugar evaporation process. *J Process Control* 15(1):67–79
36. Frank PM, Köppen-Seliger B (1997) New developments using AI in fault diagnosis. *Eng Appl Artif Intell* 10(1):3–14
37. Calado J et al (2001) Soft computing approaches to fault diagnosis for dynamic systems. *Eur J Control* 7(2–3):248–286
38. Korbicz J et al (2012) *Fault diagnosis: models, artificial intelligence, applications*. Springer, Berlin
39. Zhang J, Roberts PD (1992) On-line process fault diagnosis using neural network techniques. *Trans Inst Meas Control* 14(4):179–188
40. Svozil D, Kvasnicka V, Pospichal J (1997) Introduction to multi-layer feed-forward neural networks. *Chemometr Intell Lab Syst* 39(1):43–62
41. Stinchcombe M, White H (1989) Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In: International 1989 joint conference on neural networks, Washington DC, USA, vol 1, pp 613–617
42. Nawi NM, Atomi WH, Rehman MZ (2013) The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technol* 11(Supplement C):32–39
43. Al-Naser M, Elshafei M, Al-sarkhi A (2016) Artificial neural network application for multiphase flow patterns detection: a new approach. *J Petrol Sci Eng* 145:548–564
44. Gertler J (1998) *Fault detection and diagnosis in engineering systems*. Marcel Dekker, New York
45. Isermann R, Ballé P (1997) Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Eng Pract* 5(5):709–719
46. Venkatasubramanian V, Rengaswamy R, Kavuri SN (2003) A review of process fault detection and diagnosis: part II: qualitative models and search strategies. *Comput Chem Eng* 27(3):313–326
47. Venkatasubramanian V et al (2003) A review of process fault detection and diagnosis: part III: process history based methods. *Comput Chem Eng* 27(3):327–346
48. Venkatasubramanian V (2003) A review of process fault detection and diagnosis: part I: quantitative model-based methods. *Comput Chem Eng* 27(3):293–311
49. Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. *J Microbiol Methods* 43(1):3–31
50. Śmieja FJ (1993) Neural network constructive algorithms: trading generalization for learning efficiency? *Circuits Syst Signal Process* 12(2):331–374
51. Westreich D, Lessler J, Funk MJ (2010) Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. *J Clin Epidemiol* 63(8):826–833
52. Santín D, Delgado FJ, Valiño A (2004) The measurement of technical efficiency: a neural network approach. *Appl Econ* 36(6):627–635
53. Gurney K (1997) *An introduction to neural networks*. UCL Press, London
54. Demuth HB, Beale MH (2000) *Neural network toolbox; for use with MATLAB; computation, visualization, programming; user's guide, version 4*. Math Works
55. Reed RD, Marks RJ (1998) *Neural smithing: supervised learning in feedforward artificial neural networks*. MIT Press, Cambridge, p 346
56. Setiono R (1997) Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Comput* 9(1):205–225
57. Garson G (1991) Interpreting neural-network connections. *AI Expert* 6:46–51
58. Haykin SS, Haykin SS (2009) *Neural networks and learning machines*, 3rd edn. Prentice Hall, New York
59. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. *Trans Neural Netw* 5(2):157–166
60. Hagan M, Demuth H, Beale M, Jesús O (2014) *Neural network design*. University of Colorado, Boulder
61. Chen L (2009) Curse of dimensionality. In: Liu L, Özsu MT (eds) *Encyclopedia of database systems*. Springer, Boston, pp 545–546
62. Smith LI (2002) A tutorial on Principal Components Analysis. Computer Science Technical Report No. OUCS-2002-12. <http://hdl.handle.net/10523/7534>. Accessed 2 Feb 2018
63. Merry RJE (2005) Wavelet theory and applications: a literature study, p 41. <https://pure.tue.nl/ws/files/4376957/612762.pdf>. Accessed 20 Dec 2017
64. Dolley Shukla JS (2013) Wavelets: basic concepts. *Int J Electr Electron Eng Telecommun* 4:33
65. Mallat SG (1989) A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans Pattern Anal Mach Intell* 11(7):674–693
66. Sharma R et al (2004) Neural network applications for detecting process faults in packed towers. *Chem Eng Process* 43(7):841–847
67. Paya BA, Esat II, Badi MNM (1997) Artificial neural network based fault diagnostics of rotating machinery using wavelet transforms as a preprocessor. *Mech Syst Signal Process* 11(5):751–765

68. Banjanovic-Mehmedovic L et al (2017) Neural network based data-driven modelling of anomaly detection in thermal power plant. *Automatika* 58:69–79
69. Misra M et al (2002) Multivariate process monitoring and fault diagnosis by multi-scale PCA. *Comput Chem Eng* 26(9):1281–1293
70. Feng Z, Xu T (2011) Comparison of SOM and PCA-SOM in fault diagnosis of ground-testing bed. *Procedia Eng* 15(Supplement C):1271–1276
71. Ziani R et al (2012) Bearing fault diagnosis using neural network and genetic algorithms with the trace criterion. In: Fakhfakh T et al (eds) *Condition monitoring of machinery in non-stationary operations: proceedings of the second international conference “condition monitoring of machinery in non-stationary operations” CMMNO’2012*. Springer, Berlin, pp 89–96
72. Behbahani RM, Jazayeri-Rad H, Hajmirzaee S (2009) Fault detection and diagnosis in a sour gas absorption column using neural networks. *Chem Eng Technol* 32(5):840–845
73. Manssouri I, Chetouani Y, Kihel BE (2008) Using neural networks for fault detection in a distillation column. *Int J Comput Appl Technol* 32(3):181–186
74. Jamil M, Sharma SK, Singh R (2015) Fault detection and classification in electrical power transmission system using artificial neural network. *SpringerPlus* 4(1):334
75. Abbasi Nozari H et al (2012) Model-based robust fault detection and isolation of an industrial gas turbine prototype using soft computing techniques. *Neurocomputing* 91:29–47
76. Taqvi SA, Tufa LD, Zabiri H et al (2018) Fault detection in distillation column using NARX neural network. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-018-3658-z>
77. Kiakojoori S, Khorasani K (2016) Dynamic neural networks for gas turbine engine degradation prediction, health monitoring and prognosis. *Neural Comput Appl* 27(8):2157–2192
78. Tidriri K et al (2016) Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: a review of researches and future challenges. *Annu Rev Control* 42:63–81
79. LeCun Y et al (1998) Efficient BackProp. In: Orr GB, Müller K-R (eds) *Neural networks: tricks of the trade*. Springer, Berlin, pp 9–50
80. Starr KD, Petersen H, Bauer M (2016) Control loop performance monitoring—ABB’s experience over two decades. *IFAC-PapersOnLine* 49(7):526–532
81. Kline DM, Berardi VL (2005) Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Comput Appl* 14(4):310–318
82. Shoukat Choudhury MAA, Thornhill NF, Shah SL (2005) Modelling valve stiction. *Control Eng Pract* 13(5):641–658
83. Choudhury MAAS, Kariwala V, Shah SL, Douke H, Takada H, Thornhill NF (2005) A simple test to confirm control valve stiction. *IFAC Proc* 38(1):81–86. <https://doi.org/10.3182/20050703-6-CZ-1902.01589>
84. Jelali M, Huang B (2010) *Detection and diagnosis of stiction in control loops: state of the art and advanced methods*. Springer, London
85. Farenzena M, Trierweiler JO (2009) A novel technique to estimate valve stiction based on pattern recognition. In: de Brito Alves RM, do Nascimento CAO, Biscaia EC (eds) *Computer aided chemical engineering*. Elsevier, Amsterdam, pp 1191–1196
86. Venceslau AR, Guedes LA, Silva DR (2012) Artificial neural network approach for detection and diagnosis of valve stiction. In: 2012 IEEE 17th conference on emerging technologies and factory automation (ETFA). IEEE
87. Bacci di Capaci R, Scali C (2018) Review and comparison of techniques of analysis of valve stiction: from modeling to smart diagnosis. *Chem Eng Res Des* 130:230–265

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.