Artificial Neural Networks: A Manufacturing Engineering Perspective and Case Study Review

Eric Dimla
School of Science and Technology, RMIT University Vietnam
702 Nguyen Van Linh, District 7, Ho Chi Minh City, Vietnam
Email: eric.dimla@rmit.edu.vn; dimla@outlook.com

Abstract —This paper presents a brief review of Artificial Neural Network (ANN) application in a typical manufacturing engineering scenario. The discussion in the first part centres on the underlying principles and learning algorithms with emphasis on the basic structure of ANNs. It would be extremely laborious and tedious to list all types of neural networks herein but for the purpose of this study, an overview of those networks with proven manufacturing engineering applications was deemed necessary. The merits of ANNs and their applicability was demonstrated by reviewing work performed within the last decade in the chosen area of manufacturing engineering application, specifically Tool Condition Monitoring (TCM) in metal cutting operations.

Index Terms—ANNs, learning in ANNs, Tool wear monitoring

I. Introduction

Application of neural networks in TCM usually involves two stages: performing test cuts to acquire the necessary process data and utilisation of the obtained data post-process to investigate accuracy prediction of the network with a view to on-line or real-time application. Implementation of TCM through application of supervised paradigms involves labelling (by a teacher) of sets of data corresponding to worn and/or sharp tool states. The underlying principle in training involves the adaptation of ANN weights on each connection from neuron to neuron based on the teacher feedback. When performing this operation the differences between calculated output and expected or specified output E gives a measure of convergence success. The network is viewed as having sufficiently trained when its weights remain fixed and E is significantly small (<1%). The trained network is then capable of performing a kind of pattern recognition task upon encountering unseen sensor signal data from the cutting process either on-line or off-line [1]-[3].

The unsupervised paradigm requires no teacher and adapts its weights without feedback when an input pattern is introduced. In a metal cutting environment this method of learning would at first seem a better choice as it is quicker and eliminates the need for data labelling. Investigations by [4] showed that almost the same success accuracy prediction of tool state was achieved by both an ART-2 (unsupervised) and an MLP (supervised). They also realised that ART-2 required labelled data at the end of its implementation to distinguish tool class.

This paper provides an overview of neural networks in general, but presents a specific manufacturing engineering application example, the monitoring of cutting tool wear. In the first part of this paper, a general introduction and perspective of neural networks is presented with a brief overview of the popular architectures and learning methods. The second part then progresses to the subject of the paper, the application of neural networks in the chosen example, Tool Condition Monitoring (TCM) in metal cutting operations.

II. What is a Neural Network?

Generally, ANN have a mathematical background and theory as their development and refinement stems from basic mathematical principles modelled on biological neurons and the nervous system. Neural networks can learn and the neurons perform their operations in parallel by forming non-linear discriminants. They are robust and capable of non-linear modelling. Some of the most notable contributions have been from mathematics, biology, physiology, psychology, neuroscience and engineering, hence summarising them is difficult as it involves a wide variety of disciplines [5]-[9].

Artificial neural networks generally consists of several neurons called processing elements grouped together to form a network, performing identical tasks with the network being the main point of concern. The structure and architecture of these networks generally vary according to desired application. Ref [10] list the following as categories of ANNs:

1) Auto and Hetero-associative networks
2) Classification networks
3) Transformation networks and
4) Modelling networks

This categorisation falls short of a complete ANNs listing as no mention of prediction networks is made. The list also makes no allusion to their mutual non-exclusiveness i.e. a classifier network could be modelling as it tries to classify in the same process. In this respect it is reasonable to suggest that there are almost as many ANN as there are applications. Another school of thought classes ANN as from an architectural point of view:

Manuscript received January 14, 2019; revised July 10, 2019.
Corresponding author email: dimla@outlook.com
doi:10.12720/jcm.14.8.636-646

perceptron, associative memory and biological models [6].

A. The Structure of an ANN

Artificial neural networks take as their 'inspiration' the human brain and as such we view them as 'mere' models of the human brain. The basic information processing system of the brain is the neuron consisting of a cell body and the central nucleus. Connected to the cell body are several dendrites, which receive information from other neurons, and an axon through which the cell dispatches information to other neurons. Synaptic connection between neurons is either excitatory or inhibitory. These excitatory or inhibitory neurons fire a spike via the axon to the next neuron when the threshold value is exceeded thus communicating with other neurons. Signal transmission, be it excitatory or inhibitory, is electrical in nature in the form of ions. Learning and memory retention occurs when modifications are made to the coupling between these neurons. It is this basic operating principle that is simulated on a conventional computer to form an ANNs [11].

1) The perceptron

A perceptron is a kind of ANN commonly viewed in its simplest form as consisting of just one neuron. A more complicated version would consist of many neurons in a 'single' layer. Its multiplicative weights and biases are trainable to produce a correct target when presented with a corresponding input. A perceptron neuron or single layer perceptron network has a hard limit transfer function, an adder and a threshold function trained with the perceptron learning rule (Fig. 1).

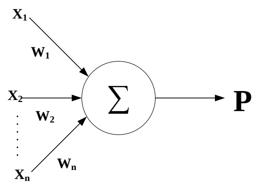

Fig. 1. Generalized diagram of a perceptron

In mathematical notation a perceptron consists of a number of neurons on a single layer. For example, if the ith weight is W_i with threshold T and its input in logical sequence Xi, then the output of the entire perceptron P is given as summarised in the following equation:

$$P = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} X_i W_i > T \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The threshold ***T*** can be replaced by a carefully chosen mapping function, the activation function such that its bounded range (0, 1) is as given by equation (1). Typically there exist many such functions. The state of the output signal is solely determined according to the summation and/or transformation of inputs Xi, multiplied by associated weights Wi. Positive and Negative weights correspond respectively to excitatory and inhibitory output effects and the performing function. Groups of more than one layer of perceptron form, what is popularly known as, the Multi-layer Perceptron.

2) The multi-layer perceptron

An NIP consists of at least three layers; an input layer, the hidden layer(s) and an output layer. The input layer links the inputs to the first hidden layer and the output layer connects the subsequent hidden layer. When data flows in one direction only, (i.e. input to output), the networks are referred to as feed-forward networks [12], [13]. Feed-forward networks therefore have a layered structure arranged such that each layer only receives inputs from the previous layers. The complexity of feed-forward MLP can be varied by increasing the number of hidden layers and nodes i.e. from simple parametric models (no hidden layer) to more complex yet flexible non-parametric models with more than one hidden layer. MLPs generally have transfer functions which perform global changes to decision surfaces.

When data flow is not in one direction only, the resulting networks are called recurrent networks. In these types of networks a closed loop exists where part of a neuron output is fed-back to its inputs thus creating recurrence. Typical examples include Hopfield, Kohonen and Adaptive Resonance Theory (ART) networks. We now briefly describe these as of networks:

3) Hopfield neural networks

These are neural network models with connections from each layer's output to that same layer's input. In certain structures the feedback connections are either direct or through several other layers. Recurrence occurs in the network whereby all the outputs are connected to the inputs creating a time-sensitive multi-layer network from only a single layer of neurons resulting in a sequential network [6], [14]. A simplified Hopfield network is shown in Fig. 2.

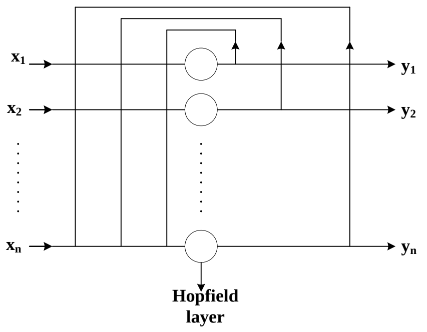

Fig. 2. Simplified diagram of a Hopfield network

4) Kohonen networks

Kohonen neural networks are most commonly referred to as self-adjusting networks (maps) because of their self-organising and adaptive mechanisms of operation. Kohonen networks were developed based on competitive learning by Teuvo Kohonen of the Helsinki University of Technology, and are constructed from fully connected arrays of neurons. They require no output data for training as the underlying information provided in the input patterns is used to form clusters or classes. Kohonen networks have two sets of weights: the computed adaptable weighted sum from the external inputs and the fixed neuron weight (Fig. 3).

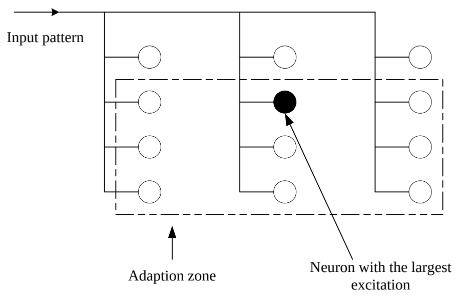

Fig. 3. A typical Kohonen feature map

5) Adaptive Resonance Theory (ART)

These networks are based on the same operating principle of self-organisation as Kohonen nets, but are much more adaptable. They were developed to model massively parallel architectures to address the problem that most neural networks are not able to learn new information on top of old information- referred to as the stability-plasticity dilemma [15]. An ART network has the ability to switch modes from plastic i.e. the learning state where internal parameters are modifiable, to the stable (fixed classification set) without detriment to previous learning.

6) Radial Basis Functions (RBF) neural networks

These networks are similar to MLP in structure (Fig. 4) being three layer structured but differ in functionality. Typically an RBF may be regarded as a form of linear non-parametric model as shown in equation (2) with an N-dimensional response. Herein, RBF changes to decision surfaces are local compared to MLP that perform global changes.

$$y_i = \sum_{i=1}^{N} W_{ij} h_i (x_{ij}) \tag{2}$$

where $h_i(x_{ij})$ is the basis function.

Its outputs are dependent upon a linear superposition of the basis functions on inputs computing radial distances to the Euclidean centres (weights vectors). The most commonly used function is Gaussian because it is capable of approximating and smoothing any arbitrary continuous function.

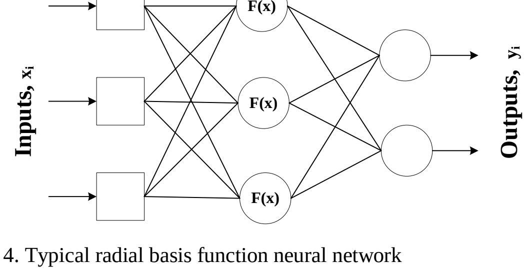

Fig. 4. Typical radial basis function neural network

Many other neural network types exist that are both popular to implement and easy to understand. These include Propagation and Counter Propagation Networks, Adaline, Bi-directional and Restricted Coulomb Energy (RCE). These networks tend to be specialist types and have not been applied in TCM systems except RCE which is notdiscussed in this paper. A comprehensive review and application of these networks can be found in [16].

When more than one neural network is combined serially or in parallel the resulting networks are called hybrid neural networks.

B. Learning

In ANN learning is the process by which weights and biases are adjusted to achieve some desired network behaviour. A parameter constant η (defined as the learning rate) controls the extent of weight and bias changes during learning. In this context the network can be viewed as having an associative memory (content addressable). Principally there are two kinds of learning methods: Supervised learning and Unsupervised learning. A third kind of learning known as Reinforcement learning is viewed as a half-way house of the former two methods.

1) Supervised learning

This is generally referred to as Hebbian learning and was the first proposed learning algorithm derived for artificial neurons. Hebbian learning in principle operates by adjusting weights proportional to the product of the outputs of pre- and post-weight neurons. Supervised learning generally involves changes in a network's weights and biases. This is accomplished by comparing the network's current output against the desired target output as specified by the tutor, resulting in a measure of the current error with the primary aim being to minimise the error.

- Perceptron learning rule

It is a learning rule used for training SLP whose operation can be summarised mathematically as follows [11]:

For all *i* and all *j*,

$$W_{(i,j)}^{t} = W_{(i,j)}^{t-1} + \eta\left(T_i - A_i\right)P_j$$
$$B_i^t = B_i^{t-1} + \eta\left(T_i - A_i\right) \tag{3}$$

where: *P*, *A* and*T* are the input, output and target vectors respectively; *W* and *B* are the weight and bias matrices respectively and η = the learning rate parameter.

The presence of hidden layers introduces additional complexity and non-linearity, which this method cannot adequately deal with. Due to such occurrences it is possible to implement higher order algorithms than illustrated by Equation (3) to train multiple layer networks. One such algorithm to have been devised is the delta learning rule.

- The delta learning rule

This method consists of making small changes to the weights by increasing or decreasing them. Every time they are increased or decreased by a fractional amount controlled by the learning parameter as defined by the following equations:

$$\Delta W_{ij}^{t} = \eta \delta X_{ij}$$
$$W_{ij}^{t} = W_{ij}^{t-1} + \Delta W_{ij}^{t} \tag{4}$$

where: η = Learning rate; δ = Correction norm of vector X; X_{ij} = Input; W_{ij} = Weight matrix.

A network undergoing delta learning performs a gradient descent search for an appropriate set of weights. Gradient descent is a mathematical method used for determining an extreme point in a curve. It requires taking repeated steps through the multi-dimensional space to the minimum. On each iteration, the direction in which the gradient vector is decreasing fastest is identified and the procedure repeated in that direction until the minimum is reached. The delta rule generalised extends to what has come to be called the Back-propagation learning rule.

- The Back-Propagation (BP) Learning

This is a learning rule in which weights and biases are adjusted by error derivative or delta vectors, back propagated through the network. Most often BP is applied only to feed-forward multi-layer networks [8], [13], [17], [18] for reasons outlined in Section 2. Back-propagation is the most popular training algorithm to have been devised. It is based on gradient descent learning being achieved as the error is affixed by propagating the output error backward through the connections to the previous layer and repeated until the input layer reached. A necessary condition is obtainable error derivatives, in effect, a continuous and bounded transfer function is desired as it satisfies the differentiability requirement.

Considering Equation (3), in order to minimise E, the weights need to be modified as defined by Equation (4). This requires the calculation of the partial derivatives of the multi-layered network thus backwardly propagating the error of the outputs [14]. Equation (4) modified to reflect this change and condition can be written as follows:

$$\Delta W_{ij}^{t} = \eta \frac{\partial E}{\partial W_{ij}}$$
$$W_{ij}^{t} = W_{ij}^{t-1} - \eta \frac{\partial E}{\partial W_{ij}} \tag{5}$$

where

$$E = \frac{1}{2}\sum_{i}(T_i - A_i)^2$$

is the Sum-Error-Square (Least Square) derived from the following cost function:

$$E0 = \left(\sum_{j=1}^{I} \left| t_j - o_j \right|^{p} \right)^{\frac{1}{p}} \quad P \in N$$

For which when P=2, simplifies to the Least Square method.

Back-propagation is generally slow in operation however methods have been devised to speed it up. The most prominent of these methods has been the addition of a momentum term μ and variation of the learning parameter [19] as shown by Equation (6).

$$\Delta W_{ij}^{t} = \eta \frac{\partial E}{\partial W_{ij}} - \mu \Delta W_{ij}^{t-1}$$
$$W_{ij}^{t} = W_{ij}^{t-1} - \eta \frac{\partial E}{\partial W_{ij}} - \mu \Delta W_{ij}^{t-1} \tag{6}$$

The momentum term technique serves to make it less likely for the BP network to be caught in shallow minima during learning. However there are still limitations in the success of this technique. More recently faster learning methods for MLP have been introduced such as conjugate gradient and quasi-Newton- thought to be superior in performance to BP [20].

2) Unsupervised learning

Kohonen suggested the first unsupervised learning method in 1984 based on ideas put forward earlier by Hebb [18]. In Hebbian learning networks weights are increased according to the product of the excitation levels of the source and destination neurons. The exact properties that unsupervised nets learn to recognise would depend on the network model and implemented learning method. Among the most prominent and often used methods are the Hopfield learning, Kohonen learning, Competitive learning and ART learning.

- Hopfield learning

This learning rule essentially emulates Hebbian learning by increasing its network weights according to the product of excitation levels of the source and destination neurons. This could be viewed as having a close association with principal component analysis with typical application networks being the Hopfield nets [11]. Hopfield learning rule begins by assigning weights to the connections before the introduction of the input pattern according to the following equation:

$$t_{ij} = \begin{cases} \sum_{s=0}^{M-1} X_i^{s} X_j^{s}, & \\ 0 & i = j, 0 \le i, i \le M-1 \end{cases} \tag{7}$$

where t_{ij} is connected weight, S is class and X is the input matrix.

With the introduction of the unknown input pattern an iterative method for convergence is applied until an output matching the unknown pattern is achieved [7].

This iterative process is mathematically shown in the equation below.

$$\mu_i(t=1) = f_h\left[\sum_{t=0}^{M-1} t_{ij}\,\mu_i(t)\right], 0 \leq j \leq M-1 \tag{8}$$

where$\mu_i(t)$ is the output of node *i* after time *t* and f_h is the hard limiting non-linearity.

- Kohonen learning

Proposed by Kohonen in the mid-1980s and based on the principle of self-organisation and adaptation, the learning rule starts by initialising the weight from input Xi to node j in time t; Wij(t), for 0 <i< n-1. Next the initial radius of the neighbourhood around node j, Nj(0) is set large. Presentation of the input pattern computes the distance dj between input and output node j using the following equation:

$$d_j = \sum_{i=0}^{n-1}\left(x_i(t) - W_{ij}(t)\right)^2 \tag{9}$$

In the meantime the weights are updated and the calculated node minimum distance dj is designated J. The new updated weights defined in the new neighbourhood size Nj(t) are then given by the following equation:

$$W_{ij}(t=1) = W_{ij}(t)\left(x_i(t) - W_{ij}(t)\right) \tag{10}$$

where $\eta(t)$ *is gain,* $0 < \eta(t) < 1$.

In effect $N_J(t)$ decreases as t $\rightarrow\infty$ thus localising the area of maximum activity [5] as shown in Fig. 6.

- Competitive learning

In this learning method individual neurons learn to become feature detectors by competing with each other for the right to respond to a given input vector and categorise input vectors among its neurons. In summary, competitive learning could be viewed as k-means clustering systems with cluster centroids as prototype vectors for some fixed number k of m group [11] as illustrated by Equations (11), (12) and (13).

Generally in competitive learning input patterns are trained to associate to a cluster of patterns. This creates an output with an indication of membership of the input pattern in a group with similar characteristics insuring a ‘winner-takes-all' situation. Two class types of competitive learning have been developed viz.: linear and differential competitive learning.

a) Linear competitive learning: Involves learning where clusters obtained from the input patterns result from output nodes competing with each other for the right to respond to the input signal. If it succeeds it outputs a '1', if it fails a '0' is output. This can be represented mathematically as follows:

$$y_i = \begin{cases} 1 \text{ if } \sum_{j=1}^{n} \mathrm{W_{ij}}x_j = Max_{k=1,\cdots \mathrm{m}}\left(\sum_{j=1}^{n}\mathrm{W_{kj}}x_j\right) \\ 0 \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{11}$$

Nonetheless a node is only allowed to learn if it can win, i.e. 'learn-only-if-you-win' or 'keep-your-distance' and is defined as shown by the equation below:

$$\frac{dw_{ij}}{dt} = y_i\left(x_i - W_{ij}\right) \tag{12}$$

Differential competitive learning: This is a technique employed and aimed specifically at stabilising the neural network system by implementing a modified version of the linear learning rule as defined above for which the modified version is shown below:

$$\frac{dw_{ij}}{dt} = \frac{dy_i}{dt}\left(x_i - W_{ij}\right) \tag{13}$$

With reference to Figure 8, exhibiting a 'winner-takes-all', the node with the largest total input claims the input pattern thus the right to respond to the input pattern (yk) i.e. yk = 1 and the rest output zeros. As the adaptation process continues (introduction of same training input pattern) output node k establishes a competitive edge over the rest. It will tend to claim the net input thus the right to respond once more, by applying the rule of thumb 'learn-only-if-you-can-win' or by virtue of it having won before. After a few cycles of adaptation the only node responding to the input pattern is yk and this means that yk dominates the network output. In the ensuing context no node can compete with yk as it would always win. A state is reached when the amount of input information flow into the system is equal to that which yk outputs thus indicating that a fully stable and trained system has been attained [15], [21].

- Adaptive resonance theory learning

The introduction of a wholly new pattern to the fully trained and stable system above may result in the adapted weights being changed hence, it would no longer re-classify or identify correctly previous patterns- the so called plasticity-stability dilemma. The plastic mode corresponds to the state when the network's weights can adapt (learn) whereas the stable state is when it would otherwise classify. If the network cannot realise when to switch modes then the output will be erroneous. A number of techniques have been devised to attempt to solve this problem. So far the most successful technique aimed at making the networks sensitive to novelty had been devised called the Adaptive Resonance Theory (ART) [15]. An ART network has a biological (cognitive) and behavioural background developed to model massively parallel processes for self-organising neural pattern recognition. They can learn (competitively) in a continually varying environment and their optimisation leaves allowance for them to be re-trained at any time or for incorporation of new training data. They are described mathematically by non-linearly coupled differential equations which are constantly changing and thus forming clusters. Adaptive Resonance Theory networks can be implemented for both real and binary, inputs, and comprise ART-1, ART-2, ART-3 and more recently fuzzy ART. Adaptive Resonance Theory network layers generally perform different functions, some possibly linked to external mechanisms with top-down and bottom-up mechanisms forming an anatomical attention-orienting system [15] as shown in Fig. 5. The ART structure is such that modified weights continually pass forwards or backwards thus cyclically resonating between the layers. Details of all the types of ART are not outlined in this review instead it is sufficient to highlight the basic algorithm for ART-1 (the first proposed ART from which the rest have stemmed).

In principle ART learning rate is adjusted according to an algorithm during training to minimise training time. Two types of learning in ART have been achieved [9]; fast learning (weight updates occur rapidly) and slow learning (slow weights update). ART learning generally operates on the same principle as competitive learning, but has the extra stability criterion for choosing winning processing elements through vigilance implementation.

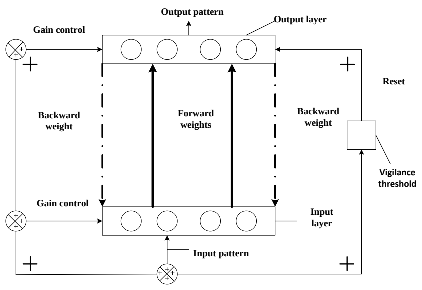

Fig. 5. Simplified diagram of ART

Broadly speaking, the ART-1 training algorithm could be divided into four stages of implementation viz. the initialisation, recognition, comparison and search phase.

In the initialisation state the top-down and bottom-up weights tij and Wij are respectively set as follows for node i and node i at time t: when tij=1, then

$$W_{ij} = \frac{1}{1+N} \tag{14}$$

for o $\leq i \leq N$-i, o $\leq$ j $\leq$ m-i, o $\leq$o _ i, with p being the vigilance parameter responsible for control of the resolution of the classification process (orienting subsystem).

In the process of a new input pattern being applied to the attentional subsystem a matching of this input is computed as follows:

$$\mu_j = \sum_{i=0}^{N-1} \mathrm{W_{ij}(t)x_i} \tag{15}$$

for 0 $\leq$ j $\leq$ M-1
where μ_j= output of node j; and x_i= input from node i (0 or 1).

The computation of the matching is preceded by choosing the best exemplar and the chosen exemplar tested. Choosing the best exemplar is carried out according to the equation below:

$$\mu_j = Max_j[\mu_j] \tag{16}$$

If the best matching is successful it is disabled and its output set to zero. Otherwise the best match is adapted using equation (17) and the whole process repeated as many times as is necessary.

$$t_{ij}(t+1) = t_{ij}(t)x_i$$
$$W_{ij}(t+1) = \frac{t_{ij}(t)x_i}{0.5 + \sum_{i=0}^{N-1} t_{ij}(t)x_i} \tag{17}$$

During training ART is sensitive to p variations. Without the intension of producing a dominating node, the initial weights must be set as low as possible.

C. *Reinforcement Learning*

Considerable differences exist between the two aforementioned learning classes in ANN that, to an extent, can be viewed as the two extreme points on the spectrum of training possibilities. In practice the network is most likely to be trained via a method that employs techniques from both methods. For systems that have feedback provided (regarding performance), but the feedback considerably weaker than should be if the training algorithm was supervised, a reinforcement learning system is instituted rather than specific target outputs [11]. Reinforcement learning schemes are enforced on a system by modifying according to a single scalar evaluation of the system's output. Ref [22] uses the classical example of stochastic learning automaton to illustrate reinforcement learning. The automaton learns and adapts by selecting from a finite set of actions in discrete time steps and the environment responds by transmitting a failure or success binary signal to the automaton. Each action response leads to the modification of the automaton state with learning effected in increased likelihood of success. The optimum conditions or best response from the automaton occurs when the likelihood probability of success is largest.

This method lacks the generalisation vital in ANN and consequently has found little or no success in terms of practical application. A successful case worth noting is the Radial Basis Function neural network which, because of its Gaussian function implication (radial symmetry property), attracts local generalisation; thereby containing both elements of supervised and unsupervised learning methods.

III. Summary of ANN Implementation

ANN has a wide range of applications- from their use in the financial markets, military target identification from sonar traces, vision systems to assist in monitoring level crossings, speech processing, to recognition and synthesis, to name a few. Industrial applications have been slow but recent areas of implementation include sensor validation and detection of tool wear, control systems and quality control, and on-line plant monitoring.

Application normally begins with a design process followed by validation and testing. Utilisation of "off-the-shelf" products (such as buying a readily made chip for character recognition) might eliminate the initial two phases. Usually, selection of desired input and output variables is decided upon first, followed by choosing suitable network architecture. With the latter no set rules exist on how many inputs, outputs and hidden nodes are required but the rule of 'conventional wisdom' abounds. Training the network proceeds architecture selection comprising finding suitable weights and bias values that lead to the best correlation between inputs and outputs by application of learning algorithms. Finally, real time test data is applied to the trained network and the network's performance evaluated.

IV. Application Example-Tool Condition Monitoring (TCM)

Applications are many and diverse with unjustifiable claims giving way to more realistic assessments of what is achievable [12], [16], [23], [24]. Some of the highly acclaimed applications include NETtalk which learns to pronounce English text; the Airline Marketing Tactician (AMT), a two stage procedure that assists airlines to forecast seat demand (first stage) and allocate airline resources to meet the projected demand (second stage); and the Electro-Cardiograph (ECG) noise filter [23]. There have also been wide applications in the financial markets to assist traders forecast the trend in bonds and stocks in Japan with remarkable successes of around 70% attainable [5], [24]. Aircraft and target identification from sonar traces are some of the military applications. British Rail developed a vision system to assist in monitoring level crossings while British Telecommunications were involved in the application of ANN for speech processing, recognition and synthesis [16], [23]. Industrial applications include sensor validation and detection of tool wear [25], [26] and shaft imbalance; control systems, authentication and quality control, automatic image analysis and on-line plant monitoring.

A. *Tool Wear Identification*

Tool wear identification tasks usually take the form (or involve at its preliminary stage), data pre-processing and recording (sampling) followed by signal analysis, commonly through application of the forward Fast Fourier Transforms (FFT) or statistical analysis (mean, kurtosis, skews, RMS). The last stage involves a decision surface or classifier using as input the processed data to reach a decision on the ensuing tool-state. A classic illustration of this process can be represented as shown in Fig. 6.

Tool condition monitoring methods based on neural networks are becoming increasingly popular because tool wear is a non-linear process and its representation by neural network is an attractive alternative to previously employed mathematical methods [27]. With ANN explicit problem description is not required and is capable of handling large amounts of data out of which it builds knowledge bases, consulted during decision making [28], [29]. Nowadays, most manufacturing systems are fast converting into fully automated environments such as is the case in Computer Integrated Manufacturing and Flexible Manufacturing Systems.

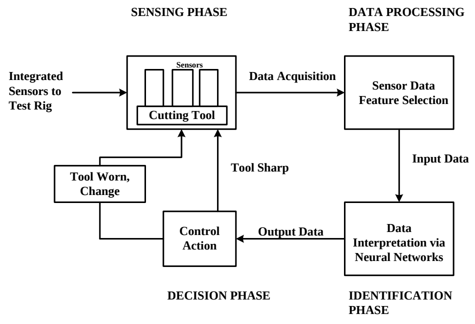

Fig. 6. Classic diagram of an ANN TCM system

Amongst the neural networks paradigms to be applied in TCM systems the most popular choice has been the Multi-Layer Perceptron neural network. Experimentation with other "s of neural networks of supervised and unsupervised nature have been pursued including less well known algorithms such as RCE network and Condensed Nearest Neighbour Network (CNNN).

B. *Utilisation of Feed-Forward MLP*

One of the earliest applications of ANN to TCM operations was proposed and performed [30]-[32]. Their primary motive was that mathematically based sensor fusion in machining processes, which were popular suffered from time consuming training, were tedious and inefficient in real-time TCM use. As an alternative approach sensor signals of Acoustic Emission (AE), cutting forces and spindle motor current were fused using NIP neural network. The pre-processed and recorded signals were first transformed using the forward FFT and then passed through a feature extraction block to select tool wear sensitive features. The extracted features were used to train (via back-propagation) and test the MLP network. The obtained results for post-process identification showed that neural networks performed better than statistically based methods.

Guillot and El Ouafi [33] using AE, cutting forces and acceleration data from a milling operation investigated three MLP network "s both having the same number of input and output nodes but a different number of hidden nodes (20, 10 and 5) to identify tool breakage. The gain and momentum of the two network topologies were set to the same values and successfully trained. Results showed that the best convergence was achieved with the network architecture having the least number of neurons in its hidden layer as it had the smallest percentage error and number of passes considered.

Yao and Fang [34] carried out turning experiments from which they recorded the cutting forces at different cutting conditions. This data was used to train MLPs with different number of neurons in the hidden layer containing different transfer functions. Firstly they trained their neural networks with different topology and obtained the RMS errors for each function as the number of neurons in the hidden node was varied. They plotted a graph of RMS errors versus number of hidden neurons for the three selected transfer functions. Results obtained clearly showed that beginning with 7 neurons in all three topologies the RMS error decreased and reached an optimum at a hidden node number of 12. Beyond this number there was a dramatic increase in RMS error. They stated that their results were inconclusive. Clearly the number of neurons in the hidden layer(s) was a crucial and complicated issue their networks were trained by BP, a method generally regarded as slow and inefficient. But nonetheless, their investigation clearly showed that it is misleading to suggest that the more the number of neurons in the hidden layer the more powerful the network.

Liu and Ko [35] using MLP structures embarked on a similar study as Ref [34] but did not limit themselves to only three neural network architectures. They investigated six MLP structures with two input nodes and an output node varying the hidden nodes from 7 to 12. After training their networks a success rate chart was made listing the percentage scored by each in identifying the tool-state. Those with <9 nodes scored up to 83.3% whereas those with >10 nodes achieved 86.6%. Liu and Ko's results when compared to the likes of [30], [31], [34] who used the same neural network architecture but not the same topology achieved fractionally lower success accuracy.

Noori-Khajavi and Komanduri [36] proposed and used an MLP as a classifier to predict drilling tool-state when data from a worn, sharp and fractured drill were introduced to the system in a block. This method can be viewed to be different from aforementioned methods involving straightforward MLP applications. It was made to classify complicated surfaces and regions using decision surfaces or boundaries introduced as class members of either a worn, sharp or a fractured tool. Their proposed network had three outputs corresponding to the three possible predictable states of the tool. They found that in real time the surface boundaries of worn, sharp and fractured tools could not linearly be separated. Members of either class encroached the boundary where it clearly did not belong and so concluded that their method was not suitable for drill wear classification.

C. Radial Basis Functions (RBF) Networks

Elanayar and Shin [27] decided to tackle the tool wear non-linearity problem on turning experiments data by the application of neural networks based on approximation theory called Radial Basis Functions (RBF). These "s of ANN are underlined by the use of a linear combination of translated functions to smoothen and approximate continuous functions. They investigated 3 different basis functions to aid in the tool wear identification process. Flank and crater wear prediction results from simulated data for the three basis functions were reported as satisfactory achieving the same success rate in each case. When test data was applied an un-quantified accuracy rate was reported which was noted to be lower compared to that obtained using the simulated data.

D. Kohonen Self-Organising Maps (KSOM)

These networks have been utilised by amongst others [28], [36]. Kamarthi *et al.* [37] chose a 20x20 grid arbitrarily and trained this with three separate sets of a 70 pattern input data from a turning test cut. Their obtained results showed that the feature map learned to correctly classify the inputs with a reliability of well over 95% accuracy. Govekar and Grabec [28] in their investigation presented an input pattern consisting of 33 components or patterns to their network. No percentage accuracy of the classification is quoted but their results indicated a high degree of success. As an extension of their work they proposed to attempt to improve their results by employing a different Kohonen training method viz. the Linear Vector Quantisation.

E. Adaptive Resonance Theory (ART)

Adaptive Resonance Theory (ART) has been used by, amongst others, [38], [39]. Burke [38] reported results similar to those obtained through application of MLP. Tansel *et al.* [39] used the sampled cutting forces from a milling operation and applied a wavelet means of transformation on them. The transformed data was then implemented through an ART-2 type neural network, and obtained results suggested that ART-2 was capable of detecting tool failure and conditions. They, however, did not quantify their success.

Choi *et al.* [40] performed TCM using both MLP and ART-2 networks (cf. [4]). In the first part they experimented with AE and cutting force data from a turning operation using a 3-layered MLP with variable number of neurons in the hidden node. Above 6 nodes in the hidden layer they reported deterioration in the correct recognition rate, scoring most successes with 4 and 6 nodes (96%). For efficiency sake they decided to choose the 4 hidden node architecture [34], and embarked upon choosing a suitable range of cutting conditions under which the monitoring system could be successfully implemented. They concluded that the system worked well for a wide range of cutting conditions provided the same cutting conditions were used off-line for training and on-line during implementation. Effectively their system was rendered specifically applicable as it was restricted to the defined cutting conditions where it was expected to perform well. For their ART-2 application they presented two groups of data to the system. Each group consisted of eight data representing worn and sharp tools. They reported that their network correctly classified and distinguished between sharp and worn tools using previously unseen tool data.

F. Restricted Coulomb Energy (RCE) Networks

Tansel *et al.* [41] used RCE to predict the tool life of a micro-drill. In their experiment they measured the micro drill thrust force which was encoded and proceeded to calculate four consecutive averages and standard deviations during each drilling operation cycle lasting between 4 and 7 seconds. RCE neural networks are parallel networks modelled on human learning and are structured much like MLP, but their arrangement allows the classification of feed forward signal in real time without special hardware. During training the hidden nodes are connected to the output nodes selectively and correspond to different pattern classes. These connections were made such that a correct output cell will be fired when an appropriate pattern class is introduced to the system. The encoded and calculated averages from 61 different drill cycles and an RCE neural network with eight inputs and two outputs to evaluate their proposed system. Initially 31 from the 61 encoded samples were used to train the network and the remaining 30 used to evaluate the trained network. The RCE network was reported to have correctly recognised 90% of the normal and tool failure cases.

G. Condensed Nearest Neighbour Networks (CNNN)

Barschdorff *et al.* [42] used two kinds of ANN in their classification experiments of cutting tool state in a turning operation: the familiar MLP and a relatively new unsupervised paradigm, CNNN. Condensed Nearest Neighbour Networks are self-organising and have been developed based on condensed clustering. Their training speeds are relatively faster than MLPs and can describe complex cluster problems. Just like MLP it has three structured layers, but differs in that its nodes in the middle layer are problem fitted (having none when untrained). As inputs to the designated CNNN Barschdorff et al. used the J-dimension feature space (patterns) of the cutting forces and AE signals. Training was performed by presentation of patterns containing two desired classes (worn and sharp). During testing if a presented pattern cannot be matched exactly to an existing (trained) cluster its nearest subclass was sought and the pattern attached to it. Obtained results showed that this newly proposed algorithm performed better than the MLP in terms of percentage success. The merits of CNNN include, amongst others, its high convergence rate and self-supervising ability.

H. Serial Combination of Network Types

Kamarthi [43] devised a novel fault-tolerant neural network model for flank wear estimation in turning by combining three types of neural networks. The first level which received the sensor signal inputs consisted of Kohonen Feature Maps, whose n-outputs were fed into corresponding RBF networks. In the final stage a recurrent network used the RBF outputs as its inputs from which flank wear was estimated. In essence [43] sought to develop a neural network structure that combined both supervised and unsupervised learning. The hybrid network reputedly offered a shorter error convergence time than the popular MLP. Turning cuts were performed and the AE, two components of vibration and three components of the cutting force were recorded. In total 3 groups of data sets were collected: the first set used for training whereas the remainder was used to evaluate the trained networks. Eight networks of exactly the same topology were designed. The first seven were trained using univariate ARMA AE, force and vibration combinations and the eighth trained using only force and vibration combination. By computing the network output results' mean, standard deviation and RMS it was concluded that their low estimation of error offered an attractive real-world application. The networks with only force and vibration used as inputs were noted to have performed better.

V. CONCLUSION

A review of the various ANN that have been applied in TCM has been carried out and presented. The review was by no means complete, as it was limited to those types, which have been applied in TCM scenarios.

The review can be summarised as follows:

- ANNs can learn from examples and generalise on unseen data
- ANNs can be regarded as universal approximators particularly suited for pattern recognition tasks.

Artificial neural networks, have matured as they have gain wider applicability. Neural networks are increasingly being applied in manufacturing processes because of their intelligent capability, and may revolutionise the way maintenance is performed.

This review has shown that the application of ANN to TCM systems tended to concentrate mainly on tool state identification and classification. As a practical aid to a tool operator, merely knowing the state of a cutting tool without knowing how well or long that tool would continue to be in that state is to say the least, unhelpful. TCM systems ought to aim at assessing and providing a real-time means of monitoring whose primary function is to help the operator maximise machine tool usage. These objectives are not attained through mere tool state classification as practical experience showed that this fails to meet the basic requirements of a TCM system. Further material relating to this can be found in ref. [26].

Practical applications of MLP such as those reviewed in this paper serve to clarify theoretical observations on matters such as best topology of an NIP for optimum performance. Results are often problem specific depending entirely on available training and test data sets.

REFERENCES

[1] D. D. E. Jnr., P. M. Lister, and N. J. Leighton, “Automatic tool state identification in a metal turning operation using MLP neural networks and multivariate process parameters,”

Int. J. of Mach Tools & Manuf., vol. 38, no. 11, pp. 343-352, 1998

[2] D. D. E. Snr, "Application of perceptron neural networks to tool-state classification in a metal-turning operation," *Engineering Applications of Artificial Intelligence*, vol. 12, no. 4, pp. 471-477, 1999

[3] W. H. Hsieh, M. C. Lu, and S. J. Chiou, "Application of backpropagation neural network for spindle vibration-based tool wear monitoring in micro-milling," *Int. J. Adv. Manuf. Tech.*, vol. 61, no. 1-4, pp 53–61, 2017.

[4] L. I. Burke and S. Rangwala, "Tool condition monitoring in metal cutting: A neural network approach," *J. Intelligent Manufact.*, vol. 2, pp. 269-280, 1991.

[5] R Beale and T. Jackson, *Neural Computing: An Introduction*, IOP Publishing Ltd, 1990.

[6] R. L. Harvey, *Neural Network Principles*, Pentice-Hall, 1994.

[7] R. P. Lippman, "An introduction to computing with neural nets," *IEEE ASP Magazine*, vol. 4, pp. 4-22, 1987.

[8] P. D. Wasserman, *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, 1989.

[9] L. Fausett, *Fundamentals of Neural Networks: Architecture, Algorithms and Applications*, Pentice-Hall, 1994

[10] J. Zupan and J. Gasteiger, *Neural Networks for Chemists: An Introduction*, VCH, 1993

[11] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Eight Reprint, Addison-Wesley Publishing Company, 1994.

[12] J. B. Gomm, D. Williams, and G. F. Page, "Introduction to neural networks," in *Application of Neural Networks to Modelling and Control, CHAPMAN and Hall*, G. F. Page *et al*., Eds., 1993, pp. 1-8.

[13] P. H. Winston, *Artificial Intelligence*, Addison Wesley, 1992

[14] R. Hecht-Nielsen, "Theory of the back-propagation neural network," in *Proc. Int. Joint Conf. on Neural Networks*, 1990.

[15] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organising neural pattern recognition machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 54-115, 1987.

[16] D. J. Myers, C. Nightingale, and Linggard (Eds.), "Neural networks for vision, speech and natural language," in *BT Telecommunications Series, Chapman and Hall*, 1992.

[17] White, *et al.*, "Artificial neural networks: Approximation and learning theory," *Blackwell*, 1992.

[18] I. Aleksander and H. Morton, "Neural Computing: An introduction," *Chapman and Hall*, 1990.

[19] A. A. Minai and R. D. Williams, "Acceleration of the back-propagation through learning rate and momentum adaptation," in *Proc. Int. Joint Conf. on Neural Networks*, 1990, pp. 1676-1679.

[20] C. M. Bishop, "Neural networks and their applications," *Rev. Sci. Instrum*., vol. 65, no. 6, pp. 1803-1832, 1994.

[21] D. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive Science*, vol. 9, pp. 75-112, 1985.

[22] P. C. Bressloff, "Stochastic dynamics of reinforcement learning," *Network: Computation in Neural Systems*, vol. 6, no. 2, pp. 289-309, 1995.

[23] ACOST, *Artificial Neural Networks*, HMSO London, 1992.

[24] J. G. Taylor, "New avenues in neural networks," in Neural Networks and their Applications, J. G. Taylor, ed., John Wiley, London, 1996, pp. 277-287.

[25] D. E. D. Jnr., P. M. Lister, and N. J. Leighton, "An investigation of the sensitivity of single layer perceptron neural networks to the inception of tool wear in a metal turning operation," in *Proc. ANNs in Engineering*, St. Louis, USA, 1996, pp. 867-873.

[26] D. D. E. Jnr., P. M. Lister, and N. J. Leighton, "Neural network solutions to the tool condition monitoring problem - a critical review of methods," *Int. J. of Machine Tools & Manufacture*, vol. 37, no. 9, pp. 1219-1242, 1997.

[27] S. Elanayar and Y. C. Shin, "Robust tool wear estimation with radial basis function neural networks," *Trans. ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 117, pp. 459-467, 1995.

[28] E. Govekar and I. Grabec, "Self-organising neural network application to drill wear classification," *Trans. ASME J. Eng. Ind.*, vol. 116, pp. 233-238, 1994.

[29] S. Markos, T. Szalay, G. Szollos, and A. W. Raifu, "Monitoring of milling processes based on artificial intelligence," *Mechatronics*, vol. 3, no. 2, pp. 231-240, 1993.

[30] S. Rangwala and D. Dornfeld, "Integration of sensors via neural networks for detection of tool wear states," *ASME PED*, vol. 25, pp. 109-120, 1987.

[31] S. Rangwala and D. Dornfeld, "Sensor integration using neural networks for intelligent tool condition monitoring," *Trans. ASME J. Eng. Ind*., vol. 112, no. 3, pp. 219-228, 1990.

[32] S. S. Rangwala, "Machining process characterisation and intelligent tool condition monitoring using acoustic emission signal analysis," Ph.D. thesis, Mech. Eng. Depart., UC at Berkeley, USA, 1988.

[33] M. Guillot and A. E. Ouafi, "On-line identification of tool breakage in metal cutting processes by use of artificial neural networks," in *Proc. Artificial Neural Networks in Engineering (ANNIE) St Louis*, USA, 1991, pp. 701-709.

[34] Y. L. Yao and X. Fang, "Assessment of chil forming patterns with tool wear progression in machining via neural networks," *Int. J. Mach. Tool Manuf.*, vol. 33, no. 1, pp. 89-102, 1993.

[35] T. I. Liu and E. J. Ko, "On-line recognition of drill wear via artificial neural networks," in *Proc. Winter Annual Meeting of the ASME, Monitoring and Control for Manufacturing Processes*, Dallas USA, 1990, vol. 44, pp 101-110.

[36] A. Noori-Khajavi and R. Komanduri, "Frequency and time domain analyses of sensor signals in drilling-11. Investigation on some problems associated with sensor integration," *Int. J. Mach. Tool and Manuf.*, vol. 35, no. 6, pp. 795-815, 1995.

[37] S. V. Kamarthi, G. S. Sankar, P. H. Cohen, and S. R. T. Kumara, "On-line tool wear monitoring using a Kohonen feature map," in *Proc. ANNs in Engineering (ANNIE)*, 1991, pp. 639-644.

[38] L. I. Burke, "Automated identification of tool wear states in machining processes: An application of self-organising neural networks," Ph.D. thesis, Depart. of Ind. Eng. and Op. Res., UC at Berkeley, USA, 1989.

[39] I. N. Tansel, C. Mekdeci, and C. McLaughlin, “Detection of tool failure in end milling with wavelet transformations and neural networks (WT-NN),” *Manufact. Sci. Eng.*, vol. 64, pp. 369-374, 1993

[40] G. S. Choi, Z. X. Wang, and D. A. Dornfeld, “Detection of tool wear using neural networks,” in *Proc. 4th World Meeting on AE and 1st Int. Conf. on AE in Manufact.*, ASNT, Boston, 1991, pp. 342-349.

[41] I. N. Tansel, O. Rodriguez, and C. Mekdeci, “Detection of tool breakage in micro-drilling operation with RCE neural networks,” *Eng. Sys. Design and Analysis*, vol. 47-1, pp. 83-88, 1992.

[42] D. Barschdorff, L. Monostori, T. Kottenstede, G. Warnecke, and M. Müller, “Wear estimation and state classification of cutting tools in turning via artificial neural networks,” in *Proc. Tooldiag'93, Int. Conf. on Fault Diagnosis*, Toulouse, France, April 5-7, 1993, pp. 669-677.

[43] S. V. Kamarthi, “Online tool wear estimation in turning through sensor data fusion and neural networks,” PhD thesis, Department of Ind. and Manufact. Eng., The Pennsylvania State University, USA, 1994.

Eric Dimla received the MEng (Hons) degree in Mechanical Engineering in 1994 from University College London (University of London) and did a PhD immediately after that on *‘Tool condition monitoring using neural networks in metal turning operations’* awarded in June 1998.

Prof Dimla is Head of School of Science and Technology, RMIT University Vietnam. Prior to joining RMIT Vietnam, he was Dean of the Faculty of Engineering at Universiti Teknologi Brunei (UTB) and Professor of Mechanical Engineering. Preceding joining UTB, he was Academic Leader (Engineering) at Southampton Solent University UK and a Senior Lecturer in CAD and Engineering at Portsmouth University, UK. Between 1998 to early 2000, he was a postdoctoral research fellow in Robert Gordon University Aberdeen Scotland, preceding his appointment as a lecturer in De Montfort University in Leicester UK. His research interest is within the area of metal cutting tool-wear and condition monitoring/fault diagnosis, intelligent sensor fusion and signal processing for industrial applications. He has published well over 50 papers in International Journals and conferences mainly in high speed machining of metals and the application of AI techniques in metal cutting tool wear monitoring.

A/Prof Dimla is a chartered mechanical engineer (CEng), Fellow of IMechE and IET as well as Fellow of the Higher Education Academy (FHEA) of the UK.