Taxonomy and a Theoretical Model for Feedforward Neural Networks

Benuwa Ben-Bright
Sch. of Comp. Sci. & Telecomm. Eng.
Jiangsu University, Xuefu Road 301
Jingkou District Zhenjiang 212013, China

Yongzhao Zhan
Sch. of Comp. Sci. & Telecomm. Eng.
Jiangsu University, Xuefu Road 301
Jingkou District Zhenjiang 212013, China

Benjamin Ghansah
Sch. of Comp. Sci.
Data Link Institute
P. O Box 2481 Tema

Richard Amankwah
Sch. of Comp. Sci. & Telecomm. Eng.
Jiangsu University, Xuefu Road 301
Jingkou District Zhenjiang 212013, China

Dickson Keddy Wornyo
Sch. of Comp. Sci. & Telecomm. Eng.
Jiangsu University, Xuefu Road 301
Jingkou District Zhenjiang 212013, China

Ernest Ansah
Sch. of Comp. Sci.
Data Link Institute
P. O Box 2481 Tema

ABSTRACT

Feedforward Neural Network (FFNN) is a surrogate of Artificial Neural Network (ANN) in which links amongst the units do not form a directed cycle. ANNs, akin to the vast network of neurons in the brain (human central nervous system) are usually presented as systems of interweaving connected "neurons" which exchange messages between each other. These connections have numeric hefts that can be adjusted and grounded on experience, enforcing adaptively on neural networks to inputs and learning capabilities. This paper presents a comprehensive review of FFNN with emphasis on implantation issues, which have been addressed by previous approaches. We also propose a theoretical model that exhibits potential superior performances in terms of convergence speed, efficient and effective computation and generality than state of the art models.

Keywords

Feedforward neural networks, Margin-Based principle, Multi-layer perceptron, Single-layer perceptron, Double Parallel Feedforward neural networks, Natural networks

1. INTRODUCTION

A FFNN is an ANN where associations amongst the nodes do not form a directed cycle. Note that this is not the same as Recurrent Neural Networks (RNNs). Further, ANNs do not have a unique single recognized definition, however, a group of statistical models may generally be christened neural if it exhibits traits as described as follows: a) Encompasses groups of adaptive numerical weights tuned by a learning algorithm, and b) Ability to approximate inputs of nonlinear functions. The adaptive weights which are the association strength amongst neurons are stimulated during prediction and training. Neural networks (NNs) are comparable to biological NNs in the performance of their roles communally and is analogous to the nodes, instead of there being a clear demarcation of sub-jobs to which respective units are allotted [1, 2]. NNs functions by creating relationships amongst processing elements instead of digital models that massages 0's and 1's during its computation. NNs are predominantly effective for forecasting events for large data array of networks based on the findings of prior studies. Stringently, a NN indicates an analogue computer, but NNs can be replicated on digital computers. In addition, the weights and the structure of these associations determines the output of the network.

In contemporary times, we have seen the application of NNs in medical imaging, industrial robotics, voice recognition systems, image recognition systems, data mining and aerospace applications. This was pioneered by Bernard Widrow of Stanford University in the 1950s [3].The modest definition of a NN, more appropriately conferred to as an ANN, is provided by Robert Hecht-Nielsen, the inventor of one of the first neurocomputers, and he explains an NN as a computing system consisting of a number of unpretentious and extremely interconnected processing elements, which execute information by their active state response to inputs externally [4, 5].

NN representations are usually recognized as an ANNs in the field of AI and are mathematically modelled by way of function definition as $f: X \rightarrow Y$or a dispersal over X or both X and Y, which are coherently linked sometimes with particular learning rulebooks or algorithm. Furthermore, in situations where the class are gotten by erratic parameters, specifics of the architecture such as the number of neurons or their connectivity and connection weights, then such instances are phrased commonly as an ANN model. The maiden and the simplest sort of type of ANN invented was FFNN, in which the flow of information is in one direction through the hidden nodes (if any) of the network to the output from the input nodes and has no loops or circles as depicted by fig. 1 below;

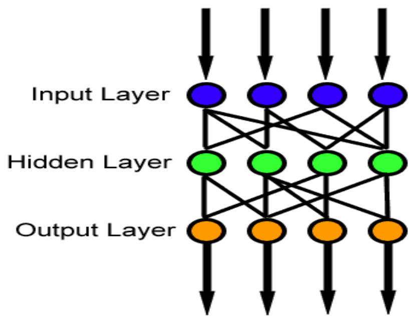
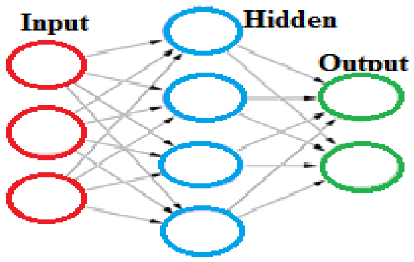


Fig 1. Structure of a FFNN

The term, "feed foreword" describes how this neural network processes the pattern and recalls patterns. neurons are normally coupled foreword when implementing FFNN and every layer of the NN holds connections to the next node. For instance, from the input to the hidden layer (HL), but however, there are no links backwards as could be seen in fig 1 above.

2. TAXONOMY OF FNN

FFNN is an inter-connection of perceptrons in which data and computations flow in a single direction, from the input data to the outputs. The number of layers of perceptrons is made up of the number of layers in a NN. FFNNs could be used to map any function from input to output and they are known as gradient based learning algorithms (Steepest Decent Method) which is the supreme algorithm used in FFNNs [6-10]. However, since FFNNs model is based on human thought processes, it is essential to appreciate how the human brain functions on a basic level [11, 12]. The brain is largely made up of cells called neurons and the human brain has about $\mathbf{10^{11}}$ of these cells which are inter-connected in a very complex network, with every neuron being connected to about $\mathbf{10^{4}}$ others [13]. These neurons can switch in about $\mathbf{10^{-3}}$ seconds, which is much longer than a switch in a computer system. Despite this, the human brain can identify a face easily; seen as a very complex computational issue in a split of a second. The ability to address complex issues in spite of neurons exhibiting comparatively sluggish switching period usually comes from the soaring degree of inter-connectedness of the neurons and an appreciable degree of parallel processing. The concerns raised above is what FFNN is seeking to model [6].



Fg 2. An ANN with interconnected groups of nodes

Fig 2. Above shows an interconnected collection of nodes, analogous to the huge network of neurons in a brain where every circular node denotes an artificial neuron and an arrow indicates a link from the output of a neuron to the input of another.

2.1 The Structure of FFNN

The building of a FFNN comprises of a (conceivably large) array of unpretentious neuron-like processing units, systematized into layers. Each unit (often called nodes) in a layer is linked with all the units in the preceding layer. These links are not all equivalent: each link may have a dissimilar forte or heft. The hefts on these links encrypt the knowledge of a network. Data enters the network through the inputs and passes through, layer by layer, until it reaches at the outputs. It acts as a classifier during normal operation, and there is no feedback amongst layers hence the reason for it being called FFNN.

In fig 3 below, a 2-layered network with, from top to bottom: an output layer (OL) with 5 units, a *HL* with 4 units, and has 3 input units respectively.

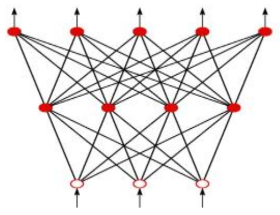


Fig 3 A 2-layered FFNN

The 3 inputs are shown as circles and these do not belong to any layer of the network (although the inputs sometimes are considered as a virtual layer by layer numeral 0). Any layer other than an OL is a *HL*. The above network thus has 1 OL and 1 HL. It further illustrates all the links among the units in various layers and a layer only links to the prior layer.

There are many ways that FFNNs can be constructed. Thus, the user need to decide on the number of neurons that will be inside the input and OLs, and also how many HLs it is going to have, as well as the number of neurons that will be in each of these HLs. There are many techniques for choosing these constraints. There are some of the universal "rules of thumb" which could be used to assist making these decisions. In nearly all cases some experimentation will be required to govern the optimum structure for the FNN.

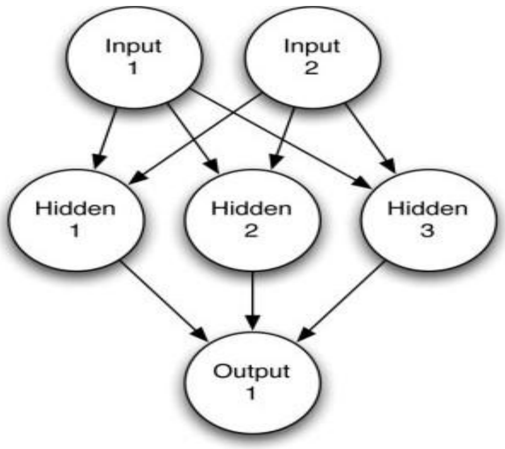


Fig 4 Conduit structure of FFNN

The input layer (IL) to the NN (as seen in fig 4 above) is the channel over which the external environment introduces an outline to the NN. So long as this outline is introduced to the IL of the NN, the OL will intend produce another pattern which in essence detains the overall performance of the NN. The condition by which the NN is trained is made known by the IL. Each input neuron is represented by some autonomous variable that has an impact over the output of the NN. The outline for the external environment is introduced by the OL of the NN. Whatever outline is offered by the OL can be unswervingly sketched back to the IL. The number of output neurons should directly relate to the type of work that the NN is to perform. To decide on the number of neurons to utilize in OL one needs to study the intended use of the NN. If items are to be categorized into groupings using the NN, then preferably one output neurons for each grouping that the item is to be allocated to is often utilized. However, if the NN is designed to carry out noise reduction on a signal then there is the possibility that the number of input neurons will tie up with the number of output neurons.

There are two choices that one considers with respect to the hidden layers are firstly the number of HLs to consider in your NN and secondly the number of neurons that will be in each of these layers? NN having two HLs can signify functions with any kind of form. There is presently no hypothetical reason to use NNs with any more than two HLs. More so, there's no need to use more than one HL for many real-world issues and problems that require two HLs are seldom bump into. Differences between the numbers of HLs are summarized in table 1 below: [7]

Table 1 Number of Neurons allowed in the Hidden Layers

Number of Hidden Layers	**Result**
None	Only capable of representing linear separable functions or decisions.
1	Can approximate arbitrarily while any functions which contains a continuous mapping from one finite space to another.
2	Represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

Again, deciding on the number of hidden neurons in layers is a very critical part in determining the overall neural network architecture. Although they do not directly relate with the external environment these layers have an incredible impact on the ultimate output. The problem of underfitting could occur when there exist too little neurons in the HLs to effectively sense the signals in a complicated data set when both the number of HLs and number of neurons in each of these HLs is not considered and could also result in the problems of overfitting and training period which are caused by using too many neurons in the HLs. Overfitting ensues when the neural network has a lot of information processing capacity that the limited amount of information confined in the training set is not sufficient to train all of the neurons in the HLs and that of the training period could occur even when there is sufficient training data. An extremely huge number of neurons in the HLs can upsurge the period it takes to train the network. The amount of training time can upsurge enough so that it is awful to sufficiently train the neural network.

Apparently, some concession must be reached among too many and too little looked neurons in the HLs and there are a lot of rule-of-thumb approaches for realizing the actual number of neurons to use in the HLs, some of which are summarized as follows.

1. The number of hidden neurons should be in the array between the size of the IL and the size of the OL in number.
2. The number of hidden neurons should be two- thirds of the IL size, in addition to the size of the OL.
3. The number of hidden neurons should be less than twice the IL size.

The three rules stated above are only preliminary points to consider. Ultimately choosing the architecture of the NN will be based on trial and error because no one will like to start throwing numbers and layers of neurons at the network randomly. To do so would be very time-consuming. There exist two methods by which the trial and error search approaches could be organized for an optimal network architecture in realizing the number of hidden neurons. These two approaches are the forward and backward selection methods.

The forward selection method commences by selecting a small figure of hidden neurons. This approach normally commences with just two hidden neurons after which the NN is trained and tested, the number of hidden neurons increased and the procedure repeated so far as the overall outcomes of the training and testing improved. The "forward selection method" is summarized in fig 5.

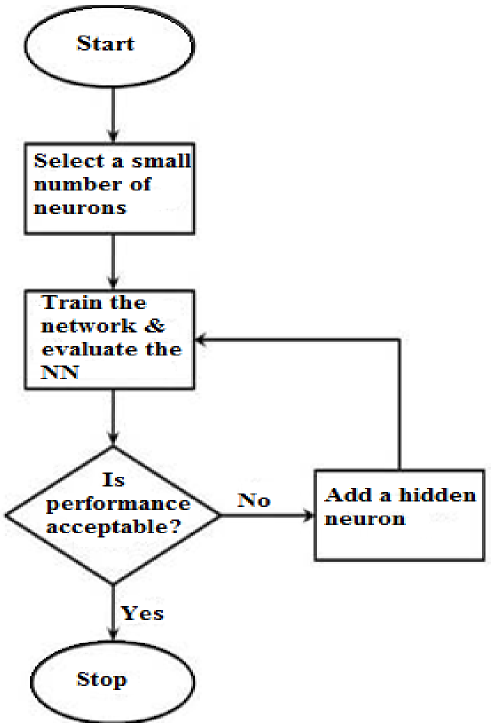

Start
Select a small number of neurons
Train the network & evaluate the NN
Is performance acceptable?
No
Add a hidden neuron
Yes
Stop

Fig 5 A flowchat of a forward selection method

The backward selection method begins by using a huge number of hidden neurons. Then the NN is trained and tested. The procedure lasts until about the performance perfection of the NN is no longer significant.

One supplementary technique that can be used to lessen the number of hidden neurons is known as pruning, this encompasses assessing the weighted links among the layers. If the network contains any hidden neurons which holds only zero weighted links, they can be aloof which makes pruning a very important concept NNs. [21]

2.2 Operation of FFNN

The operation of FFNN could be categorized into two phases: the learning phase, and the classification phases.

In the learning phase, FFNN utilizes a supervised learning algorithm. Furthermore, the NN needs to know the groupings with which the pattern should be referenced aside the input design, and the learning proceeds as follows: a pattern is offered at the inputs and is transformed in its passage over the layers of the network till it gets to the OL. The outputs of the network (with units all belonging to a different category) are then sampled with the outputs as they would preferably have been if this pattern were correctly classified. In the latter case the unit with the right category would have had the biggest output reading and the output readings of the other output units would have been very small [14]. In view of the above analysis, all the connection weights are altered a bit to pledge that, the next time this or a similar pattern is placed before the inputs, the value of the output unit that corresponds with the precise categorization is slightly higher than it is now and that, at the same time, the output values of all the other inappropriate outputs are a bit lesser than they are now [15].

The variances that exists between the real outputs and the ideal outputs are propagated backwards from the top layer to lower layers to be utilized at these layers to amend connection weights and NNs of these sorts are often described by the term backpropagation network. The period for learning phase is heavily dependent on the size of the NN, the number of epochs, the number of patterns to be learned, the tolerance of the minimizer and the speed of your computer will tell how much computing time the learning phase may take. Backpropagation is the utmost implemented training technique commonly used for FFNNs. Its principal objective is to offer a machinery for apprising connected neurons grounded upon minimization of error. To accomplish this, gradient descent is generally used to establish the steepest path toward the minimum of

$$E\,\vec{w} = \frac{1}{2}\sum_{d\in D}(t_d - O_d)^2 \qquad (1)$$

where a training instance in D is **d**, $\mathbf{t_d}$ is the target value, $\mathbf{O_d}$ is the output value, and $\vec{w}$ is the weight vector. Backpropagation entails determining an error by foremost feedforwarding (FF) inputs into the network and deducting the outcome from some target output. This variance is then bourgeoned by the derivative of the neuron's activation function and as in the case of sigmoid, as shown in equation (2) below, and stored for reference by the update at the prior layer.

$$f^{'}(net) = O(1-0) \qquad (2)$$

Advancement is further made to calculate the errors layer by layer, traversing backwards through the network and performing the neuron's error computation, which is the derivative of the neuron stimulation function bourgeoned by the sum of every output weight's multiplication with the forward neuron's error term. Thereafter every error term is calculated, we update the weights by the multiplication of each branch's output with the forward node's error and the learning rate.

In the classification phase, the hefts of the network are fixed. A pattern, is transformed from one layer to the other till it gets to the OL when presented at the input layer. The classification can result by choosing the grouping related with the output unit that has the largest output value. In contrast to the learning phase classification is very fast.

2.3 Single - Layer Perceptrons

The modest kind of NN is a single-layer perceptron network, which comprises of a single layer of output nodes; the inputs are nourished directly to the outputs through a sequence of weights and maybe regarded as the simplest kind of FFNN. The summation of the products of the weights and the inputs is determined in each node, and in case the outcome is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons exhibiting this kind of activation function are also better known as linear threshold units or artificial neurons. In the literature the term perceptron often recognized as networks consisting of just one of these units and similarly described as neuron by Warren McCulloch and Walter Pitts in the 1940s [16, 17].

A perceptron could be formed by utilizing any of the activated and deactivated states values so long as the threshold value lies between the two states. Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks formed from nodes with different deactivation and activation values. Perceptrons can be trained by simply learning and implementing the delta rule algorithm that is capable of determining the errors between sample output data and, calculated output and uses this to create a modification to the weights, hence employing a form of gradient descent [18]. Single-unit perceptrons only have the capability of learning linearly separable patterns demonstrated in the well-known monograph entitled Perceptrons, by Papert and Minsky to indicates that it was impossible for a single-layer perceptron network to learn an XOR function and further proposed that a similar result would hold for a multi-layer perceptron network [19]. However, this is not true, as both Minsky and Papert recognized that multi-layer perceptrons were capable of producing an XOR Function as shown in fig 6 below [20].

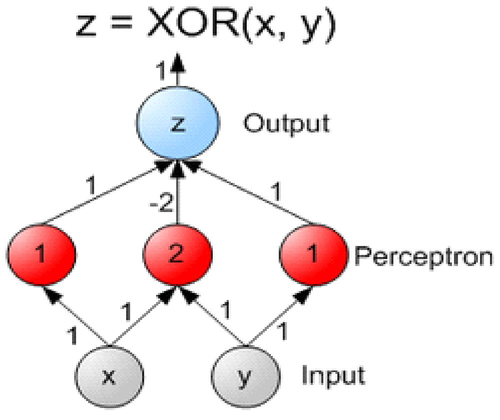


Fig 6: A three-layer Perceptron net capable of calculating XOR.

The computational power of the network is quite not unlimited despite the fact that it relies on a single threshold unit and it has been shown that any continuous function could be approximated by networks of parallel threshold units from a compacted interval of real numbers into the interval [-1,1] [21]. A continuous output could be gotten out of a multi-layer NN other than a step function which is seen in the logistic function:

$$y = \frac{1}{1+e^{-x}} \qquad (3)$$

The f(X) which normally replaces x, where f(X) is an analytical function in terms of x. in view of this. the single-layer network is comparable to the logistic regression model, extensively utilized in statistical models. The logistic function of which is also referred to as sigmoid function which has an unremitting derivative, and makes it suitable for backpropagation [22]. It is preferred ideally because its derivative is easily determined according to the chain rule [23]:

$$y' = y(1-y)\frac{df}{dX} \qquad (4)$$

2.4 Multi- Layer Perceptrons

This class of networks which comprises of multiple layers of solvable units, generally inter-connected like a FFNN. Every neuron in a layer has directed links to the neurons of the succeeding layer. In numerous implementations, a sigmoid function is utilized by the units of these networks as an activation function [24].

The universal approximation deduction for NNs states that every unremitting function that joins intervals of real numbers to some output interval of real numbers can subjectively be approximated carefully by a multi-layer perceptron with just a HL. This outcome embraces for an extensive range of activation functions, such as the sigmoidal functions [25].

Multi-layer networks adopt a diversity of learning procedures, the most common one being back-propagation, in which the output results are likened with the correct answer to determine the result of some predefined error function. More so, the error inputted back into the network through numerous techniques, and the weight of every link is adjusted with the algorithm using the information of the network, which in effect reduces the value of the error function by some minute amount. The network will typically converge to some state where the error of the computation is small, after the afore process is repeated for a satisfactorily huge number of training sets which could be seen as the network learning a particular target function. To adjust weights properly, one applies a gradient descent, a general method for non-linear optimization, and the derivative of the error function is determined with respect to the weights of the network. However, they are changed to augment decreases in the error (accordingly going effortless on the outward of the error function). For this reason, backpropagation can only be useful on networks with differentiable activation functions.

The problem of teaching networks to perform well, in particular on samples that were generally not used as training samples; this is usually a daunting task that requires extra techniques, especially for cases where limited number of training samples are presented [26, 27]. The risk associated with it is that, the training data is usually over fitted by the network and it fails to record the actual statistical method engendering the data[28]. Computational learning concept is interested with training classifiers on an inadequate amount of data. In the context of NNs a modest exploratory, called early stopping, usually guarantees that the network will take a broad view on examples that are not part of the training set [29]. Other distinctive issues with the backpropagation procedure are the convergence speed and the likelihood of culminating up in a local minimum of the error function. Today there are real-world approaches that make backpropagation in multi-layer perceptrons the instrument of interest for many machine learning tasks.

The contributions of the paper include a proposed novel training principle for FFNNs centered on Margin-Based Principle (MBP) for FFNNs and that of the Double Parallel Feedforward Neural Networks (DPFNN) to further improve the convergence rate and the generalization capabilities of FFNNs in addition to an algorithm to deal with the issues concerned with optimization and subsequent improvement in the performance of learned policy as well as make it less computationally expensive.

The rest of the paper is structured as follows; we review related methods of FFNN in section III. Section IV discusses our general framework to FFNN, we look at applications of our proposed technique in section V, and section VI presents summary and future direction.

3. RELATED WORK

According to [30] FFNN is a traditional classifier which is very popular at present just like deep architecture of neural network (DNN) in terms of application and theory as extensively reviewed by [31]. However, the training algorithm of FFNN, no matter the shadow or the deep, is based on the Widrow-Hoff Principle (WHP) that minimizes the squared error with some weight regulation items [32]. This kind of learning algorithms need lots of labelled samples and tend to be overfitting. The general capability of these NNs is limited by the least square error optimization procession and, WHP so [30] employed the MBP which could handle some few labelled samples and would be seen as a sparse learning structure instead of WHP to acquire a better generalization ability, which motivated their novel model. The perceptron that could be treated as linear classifier, shares the disadvantages of NNs, since it also needs lots of labelled data and is tend to be overfitting.

When the Max-Margin Principle is applied to perceptron or linear classifier, it overcame the disadvantages of WHP, thus it tolerates less labelled data and gained better generalization ability. The afore-mentioned observations inspired the researchers to, apply the MBP to the NN training process that could achieve better accuracy with less labelled data and a novel learning model was proposed and could abandon part defects of traditional FFNNs. In order to address the issue, FFNN was treated as a two-step process. The process of IL to hidden layers was treated as feature abstraction and the process of hidden layers to OL was treated as classification.

The FFNN makes use of linear regression learners to abstract the samples to a new feature space, thus by applying the Min-Margin Principle to the feature abstraction process to minimize the error of regression or error of abstraction to improve the ability of feature abstraction. While in the classification process, FFNN makes use of linear classifiers to discriminate, Max-Margin Principle was used to minimize the structural risk. The two processes were combined to an optimization problem which was used as a model.

In the quest to improve on the convergence rate and the generalization capabilities of FFNNs, [33] proposed a DPFNN which parallelly combines a connection of a multi-layer FFNN and a single layer FFNN as indicated in fig 7 below.

As a result, some weak and strong convergence results are obtained, indicating that the gradient of the error function tends to zero and the weight sequence goes to a fixed point, respectively. The figure below depicts DPFNN

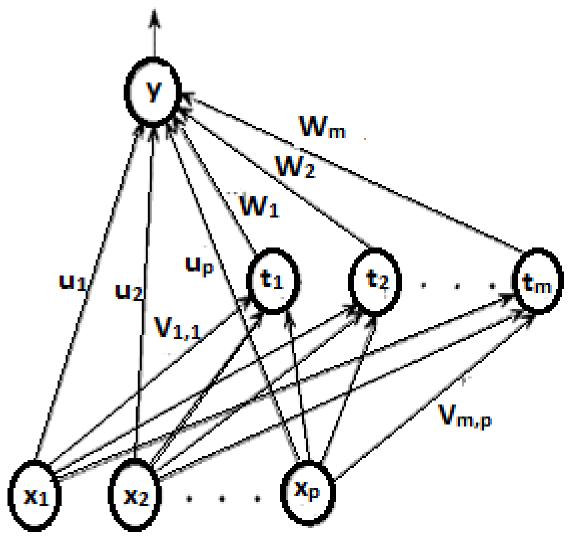


Figure 7. Topological Structure of DPFNN

The most widely used learning method for DPFNN remains to be the gradient method [33-36]. It is shown by[37] that the training speed and accuracy are greatly improved for DPFNN compared with corresponding multi-layer feedforward neural networks [38-43]. A double parallel feedforward process neural network with similar structure and updating rule as DPFNN is proposed in[44]. In [45], an alternate learning iterative algorithm for DPFNN is presented. The truncation error caused by word length on the accuracy of DPFNN is analyzed in[46].

NN is one of the early AI models, and now becomes a large branch of learning algorithms. The paper's focus was on the famous feed-forward back prorogation classification networks with non-linear mapping. It stated the three kinds of ways to popularly promote FFNN; to add weight regularity item to the optimization target, to prune the surplus links and to go into deep architectures. The first way is to refine the weights of the NN and the other ways are to revise the structure of it. [47] had studied the effect of weight decay and states that it has two effects, suppressing any irrelevant of the weights and improving the generalization ability. [48] had studied the pruning of RBFFNN, which firstly introduces the concept of significance of the hidden neurons and then uses it in the learning algorithm to realize parsimonious networks. Recently such stated in [49] , going to deep catches many eyes, since not only just adding the hidden layers could gain an improvement in performance, but DNN can also automatically select features and amazingly complete the comprehension missions. [50] had applied deep network into natural languages, and many works such as [51, 52] had applied deep network into image processing, deep learning is one of the hottest topic in today's AI. Before [53] and [54] proposed the fast unsupervised or supervised methods, multi-layer NNs are hard to train, this kind of difficulty is analyzed in [55] and [56], for the reason that the optimization process is often stuck into the local optima. The work stated above is a contribution independent from deep learning, or a principle, which could both work for shadow or deep architecture of FFNNs. As to the difficult of multilayer training algorithms which is applied with the stated principle, it could also be solved by the same kind of deep learning tricks introduced. Deep learning is a kind of contribution to the architecture of NNs and this paper is a kind of contribution to the training principle. Deep architecture and our principle could be merged to generate a new powerful model that should be more competitive in terms of performance than these two methods in isolation. The famous Margin-Based model, which is a Support Vector Machine (SVM) was proposed by [57]; SVM could have many advantages such as small sample learning and sparse learning structures. However, MBP could be applied to many models to achieve the similar benefits of SVM. [58] had applied the Max-Margin Principle into the Markov Networks, and[59] had applied the Max-Margin Principle into classification of data with absent features. Both of these two works belong to supervised learning. [60] Had applied MBP into feature selection and [61] had applied Max-Margin Principle into Clustering. Both of these two works belong to unsupervised learning[62]. Recently, [63] had introduced this principle to on-line learning for Markov Logic Networks, and [64] had introduced it to early event detection. As above, MBP and DPFNN principles could be applied into FFNNs, thus a critical outlook and contribution of this paper. This work could be applied into NNs together with weight decay, link pruning and deep architectures.

The FFNN could be treated as two processes as in the case of current studies; the abstraction process and classification process. Take two-layer FFNN as an example, the process of IL to hidden layers corresponds to abstraction process. In this stage, each hidden neuron is treated as a linear regression learner to fit some parts of the data, and hidden neurons deal with the regression results with non-linear function to get its output. In this process, the features of the samples could be converted into new space, where the hidden neurons play a role as the basis. The process of hidden layers to OL corresponds to classification process. In this stage, each output neuron is treated as a linear classifier to discriminate different classes of data, and output neurons also deal with the classification result with non-linear functions to get its output.

The IL to HLs is the abstraction process where we could apply the Min-Margin Principle, and the HLs to OL is the classification process where we could apply Max-Margin Principle. It means that all the HLs could be an abstraction process where Min-Margin Principle works and the last layer that corresponds to the output could be a classification process where Max-Margin Principle is applied. The FFNNs have HLs each of which has number of hidden neurons and a number of output neurons, with a number of labelled samples. So, we combine the two principles in one optimization problem, as maximizing. [65] Proposes a new MBP centered discriminative feature learning approach that specifically aims at learning a low dimensional feature representation to maximize the global margin of the data and samples from the same class as close as possible. [66] Also recently proposes a performance evaluation of loss functions for speech recognition to enhance the generality of acoustic model by implementing an MBP

A more recent work also looked at training efficiency and computational cost, both in general-purpose and in embedded computing environments and a strategy to convert a network configuration between different activation functions without altering the network mapping capabilities in FFNN were also presented in [67]. These literatures serve as a stimulant for this paper.

4. DISCUSSIONS OF THE THEORITICAL FRAMEWORK

Gradient based systems normally are prone to (1) over-fitting, (2) Easy to convergence to local minima, (3) Converges slowly, and (4) difficult select learning rate despite numerous attempts by past studies to address these challenges we still have not arrived at an optimal solution. The aforementioned draw backs could be overcome by reducing the network complexity and also by increasing the problem complexity of the network. Despite the merits and popularity of backpropagation learning there are some fundamental problems which deserve further attention. First, supervised learning using a backpropagation or a backpropagation-like gradient-based learning rule can be stuck in a local minimum of the error function in the sense of local minima due to the gradient descent nature of gradient-based learning rules. Because local minimum errors are still potential pitfalls undermining or plaguing supervised learning, further investigations are deemed necessary. Backpropagation-like procedures require a large number of computations per iteration so that the algorithm runs slowly unless implemented in expensive custom hardware. Moreover, the procedures need much iteration to converge and, thus, are inappropriate for on-line real time, rather than off-line learning.

Feed forward architecture with a single layer [68] and multiple layer[40, 43, 69-71] can be used as universal approximator given mild assumptions on hidden layer but the rule excludes backward connections as could be seen in recurrent networks[72-76]. More so, in FFNN no memory or delay is allowed, which makes the network only useful to represent static, models[77]. The two most important issues realized which needs to be addressed are training efficiency and computational cost. The FFNN suffers from slow learning speed due to its backpropagation strategy which is as a result of its rule for the computation for weights correction matrix, calculated using the derivative of the activation function for the neurons. The universal approximation theorem[68] states that one of the conditions for the FFNN to be a universal approximator is for the activation function to be bounded. For these reasons, most of the activation functions show a high derivative near the origin and a progressive flattening moving towards infinity. We believed that problems enumerated above could be solved by the use of activation functions which will helps reduce the computation cost; the margin-based principle could also be combined with the DPFNN algorithm to further deal with the problem of over-fitting, slow to convergence rate as well as convergence to local minimal problems thereby increasing the problem complexity of the network. The problem of slow convergence rate could be addressed by a combination of FFNN with fuzzy logic (FL) to create fast convergence [78].

The abstraction process which is the process of ILs to hidden layers, makes use of linear regression learners to predict/fit different characteristics of samples which is optimized by the least square errors as stated below;

$$\min \quad \mathrm{Z} = (<\vec{w},\vec{x}> -v)^2$$

But the margin-Based Principle minimizes the margin instead of the distance; hence Min-Margin Principles takes the minimum distance from the samples to optimize the target as follows;

$$\min Z = \frac{(<\vec{w},\ \vec{x}>)v}{|\vec{w}|}$$

From the equation y is the label of the sample for some linear abstraction learners. The principle stated above is geometric invariant and more essential as one of the reasons for the promotion of the abstraction process.

The classification phase which is the process from the hidden layers to the output could make use of linear classifiers to discriminate the different classes. Traditionally, this could be stopped at any suitable position in a linear hyper – plane and could result in overfitting problem. This structural risk is effectively addressed by the Max-Margin Principle and thus improves the generalization ability for the optimization problem as follows;

$$\max Z = \frac{(<\vec{w},\ \vec{x}>)v}{|\vec{w}|}$$

The FFNN with H hidden layers each of which has H_h hidden neurons and N output neurons, the number of labelled samples is L. So, the two principles are combine in one optimization problem, as maximizing and is formulated as the objective function below;

$$Z = \sum_{t=1}^{L}\{\sum_{i=1}^{N}\frac{<w_i^{\to out},v^{H+1}>t_i}{|w_i^{\to out}|} - \lambda\sum_{m=1}^{H}\sum_{j=1}^{H_h}\frac{<w_j^{\to m},\ v^{\to m}>v_j^{m+1}}{|w_j^{\to m}|}\} \quad (5)$$

Where $w_i^{\to out}$ is the weight vector from last hidden layer to the i-th neuron in OL, and the $w_j^{\to m}$ is the weight vector from $(m-1)$-th hidden layer to the i-th neuron in m-th hidden layer. t_i is the output of i-th neuron in OL, and v_j^m is the output of j-th neuron in m-th HL. $\vec{x}_t$ is the input vector, and $\mathrm{v}^{\vec{m}}$ is the output vector of (m − 1)-th HL. $\mathrm{v}^{\vec{m}}$ is composed by v_j^m . λ is an hyper-parameter in the training algorithm. The first term of the above formular represents Max-Magin Principle and the second term represent Min-Margin principle. However, since the output of a nonlinear function must be symmetric for both maximum positive and minimum negative values as in equation (6) and (7), the training algorithm adopts the gradient ascent for the optimization target as seen below;

$$\sigma(x) = \frac{1}{1+\exp(-x)} - 0.5 \quad (6)$$

$$\sigma'(x) = \big(0.5+\sigma(x)\big)(0.5-\ \sigma(x)) \quad (7)$$

$$Z_t = \sum_{i=1}^{N}\frac{<w_i^{\to out},v^{H+1}>t_i}{|w_i^{\to out}|} - \lambda\sum_{m=1}^{H}\sum_{j=1}^{H_h}\frac{<w_j^{\to m},\ v^{\to m}>v_j^{m+1}}{|w_j^{\to m}|} \quad (8)$$

We could work out the partial derivative of the target to express this formula in a brief way, by introducing a δ function and it works out iteratively to result in equation (9) below;

$$\delta_{j,m-1}^{m} = \sum_{s=1}^{Hh}\delta_{s,m}^{m}\ w_s^{\to m}(j)\delta'(<w^{-m-1},v^{\to m-1}) \quad (9)$$

Then, we define a γ function.

$$\gamma_j^m = \frac{v_j^{m+1}}{w_j^{\to m}}v^{\to m} - \frac{<w_j^{\to m},v^{\to m}>v_j^{m+1}}{|w_j^{\to m}|^3}w_j^{\to m} + \frac{<w_j^{\to m},v^{\to m}>\sigma'(<w_j^{\to m},v^{\to m}>)}{|w_j^{\to m}|}v^{\to m} \quad (10)$$

With the form of δ function, we could reduce the computational complexity and express the gradient, very briefly.

$$\frac{\partial Z_t}{\partial_t^{\to out}} = \frac{t_i}{|w_i^{\to out}|} v^{\to H+1} - \frac{t_i < w_i^{\to out}, v^{\to H+1} >}{|w_i^{\to out}|^3} w_i^{\to out} \quad (11)$$

$$\frac{\partial Z_t}{\partial w_t^{\to m}} = \delta_{i,m}^{out} v^{\to m} - \leftthreetimes \sum_{s=m+1}^{H} \delta_{i,m}^{s} v^{\to m} - \leftthreetimes \gamma_j^m \quad (12)$$

In above formula, σ′ is the derivative of the non-linear function. With the derivative of the target, we can obtain the updating equation as below.

$$w_i^{\to out} = w_i^{\to out} + \alpha * \frac{\partial Z_t}{\partial w_t^{\to out}} \quad (13)$$

$$w_j^{\to m} = w_j^{\to m} + \alpha * \frac{\partial Z_t}{\partial w_t^{\to m}} \quad (14)$$

α is the learning rate. So, the training algorithm is achieved. This method in some special case may be stuck into local optima and is a common problem for neural networks.

The gradient decent method could be further enhanced by introducing the Double Parallel procedure with P input nodes with 1 output nodes into the structure of the FFNN. The weight vector connecting the HL and the OL is denoted by $w = (w_1, w_2, \dots, w_m)^T \in R^m$ and the weight matrix connecting the IL and the HL by $V = (v_{i,j})mxp$ where $v_i = (v_1, 1, \dots, v_{i,p})^T \in R^p$ is the weight vector connecting the IL, and *i*-th node of the HL. Similarly, we denote the weight vector connecting the ILand the OLby $U = (u_1, u_2, \dots, u_p)^T \in R^p$

All the weight vectors are incorporated into one weight vector as

$$W = (u^T, v_1^T, \dots, v_m^T, v_w^T)^T \epsilon R^{p+mp+m}$$

For a giving set of training samples supplied to the neural network, the error function is defined as

$$E(W) = \frac{1}{2} \sum_{j=1}^{Z} (v^j - O^j)^2 \quad (15)$$

Where v is the actual output of the neural system, O is the highest sample and the purpose of the network learning is to find W* such that

$$E(W^*) = \min E(W) \quad (16)$$

The batch learning approach was followed by this paper to get the partial derivative of the error function E (W) for the respective weight vectors. After which they are refined by an iteration process and the learning rate, η at each stage of the iteration must be greater than zero. We then substitute this into equation (15) above. This weight vector we introduced at every stage of our first module. Hence equations (8), (9), (10), and (12) are updated as (17), (18), (19), and (20) respectively;

$$Z_t = \sum_{i=1}^{N} \frac{< w_i^{\to out}, v^{H+1} > t_i}{|w_i^{\to out}|} - \rightthreetimes \sum_{m=1}^{H} \sum_{j=1}^{H_h} \frac{< w_j^{\to m}, v^{\to m} > v_j^{m+1}}{|w_j^{\to m}|} + \eta \sum_{j=1}^{T} (v^j - O^j)^2 \quad (17)$$

$$\delta_{j,m-1}^{m} = \sum_{s=1}^{Hh} \delta_{s,m}^{m} w_s^{\to m}(j) \delta'(< w^{-m-1}, v^{\to m-1} > +\eta \sum_{j=1}^{T} (v^j - O^j)^2 \quad (18)$$

$$\gamma_j^m = \frac{v_j^{m+1}}{w_j^{\to m}} v^{\to m} - \frac{< w_j^{\to m}, v^{\to m} > v_j^{m+1}}{|w_j^{\to m}|^3} w_j^{\to m} + \frac{< w_j^{\to m}, v^{\to m} > \sigma'(< w_j^{\to m}, v^{\to m} >)}{|w_j^{\to m}|} v^{\to m} + \eta \sum_{j=1}^{T} (v^j - O^j)^2 \quad (19)$$

$$\frac{\partial Z_t}{\partial w_t^{\to m}} = \delta_{i,m}^{out} v^{\to m} - \leftthreetimes \sum_{s=m+1}^{H} \delta_{i,m}^{s} v^{\to m} - \leftthreetimes \gamma_j^m + \eta \sum_{j=1}^{T} (v^j - O^j)^2 \quad (20)$$

Where η is assumed to be 0.5 in this case.

4.1 Fast Computation Using Activation Functions

There are two costs associated with the computational cost. The first one is a linear cost, emanating from the operations needed to perform the sum of the weighted inputs of each neuron. The second one related to the computation of the activation function is nonlinear. In a computational environment, those operations are carried out considering a particular precision format for the numbers. Fixed-point and floating-point arithmetic are the most commonly used to compute the FFNN elementary operations. The linear part of the FFNN is straightforward; operations of products and sums.

Three branches of these approaches can be found in literature; PWL (piecewise linear) interpolation, used to carry out in embedded environment since, for every segment, the approximated value can be computed by one multiplication and one addition. In [79] an implementation of a NN on two FPGA devices by the Virtex-5 Xilinx, and the Spartan 3 was proposed. LUT (Lookup-Table) inter-polation is the modest method that can be castoff to lessen the cost of the computational of the complex function. The idea is to stockpile inside the memory (a table) samples from a subdomain of the function and access those instead of calculating the function, and higher order/hybrid techniques which combines both LUT and PWL approximations to acquire a result that produces a concession between the speed of the LUT and the accuracy of the PWL approximation.

5. APPLICATIONS

FFNN, a surrogate of ANN, has a comprehensive applicability to real world business issues. In fact, they have already been positively applied in many industries. Since the network is best at recognizing trends in data, it is well suitable for forecasting needs including: industrial process control, sales forecasting, data validation, customer research, risk management, and target marketing.

It is also utilized for the following explicit paradigms: interpretation of multi-meaning Chinese words; analysis of hepatitis; recognition of orators in conversations; salvage of telecommunications from defective software; undersea mine detection; texture scrutiny; three-dimensional object recognition; hand-written word recognition; and facial recognition.

It is a popular research area in the field of medicine and is believed to receive a widespread application to biomedical systems in the coming years. This is due to the fact that it is perfect in identifying ailments using scans, since there is no need to offer a precise algorithm on how to detect the disease. It applies a learning approach that mimics established examples thereby shielding details of illness. What is desired is a set of examples that are illustrative of the various facets of the disease. Note that, the quantity of examples is not important. The examples ought to be carefully chosen if the system is to perform efficiently and reliably [80].

6. CONCLUSIONS

In this paper, we have presented a comprehensive review on FFNNs, we looked at the MBP into the training algorithms of FFNNs, and also various models capable of handling sparser labelled datasets and high-dimension datasets with high accuracy, while modification from old ANN method to our proposed method is efficient and easy to work with. It was also observed that DPFNN has a faster convergence speed and better generalization capability than standard FFNN. However, our work outperformed prior state-of-the-art works with regards to high convergence rate, and also has an additional capability of handling more datasets which are computationally less expensive. Naturally the FFNN would have had a saturation problem with the sigmoid function, and for this reason, the training is hardly possible in high-dimension dataset. However, the technique implemented in this solves this problem, since our model does not involve the derivative of sigmoid function. The model could solve high-dimensional problems directly and naturally while prior techniques must use other methods. This is one of the major advantages of our proposed model. However, investigations into using diaphone-level acoustic models, sparse representations and deep learning is of interest as this would allow state-sized representations of speech to be used instead, which may further enhance performance.

7. REFERENCES

[1] A. Kumar and F. Shaik, "Image Processing Methods Utilized," in *Image Processing in Diabetic Related Causes*, ed: Springer, 2016, pp. 9-18.

[2] D. J. Larimer, "Processes and Systems for Automated Collective Intelligence," ed: Google Patents, 2007.

[3] R. Cutler and A. Kapoor, "System and method for audio/video speaker detection," ed: Google Patents, 2008.

[4] M. Caudill, "Neural nets primer, part VI," *AI Expert,* vol. 4, pp. 61-67, 1989.

[5] R. Hecht-Nielsen, "Neural network primer: part i," *AI Expert,* pp. 4-51, 1989.

[6] W. Wu, G. Feng, and X. Li, "Training multilayer perceptrons via minimization of sum of ridge functions," *Advances in Computational Mathematics,* vol. 17, pp. 331-347, 2002.

[7] W. Wu, G. Feng, Z. Li, and Y. Xu, "Deterministic convergence of an online gradient method for BP neural networks," *IEEE Transactions on Neural Networks,* vol. 16, pp. 533-540, 2005.

[8] W. Wu, N. Zhang, Z. Li, L. Li, and Y. Liu, "Convergence of gradient method with momentum for back-propagation neural networks," *JOURNAL OF COMPUTATIONAL MATHEMATICS-INTERNATIONAL EDITION-,* vol. 26, p. 613, 2008.

[9] W. Sun and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming* vol. 1: Springer Science & Business Media, 2006.

[10] Z. Li, W. Wu, and Y. Tian, "Convergence of an online gradient method for feedforward neural networks with stochastic inputs," *Journal of Computational and Applied Mathematics,* vol. 163, pp. 165-176, 2004.

[11] R. C. O'Reilly and Y. Munakata, *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*: MIT press, 2000.

[12] D. Fagan, "JMuTeaches its last course," *chance,* vol. 63, p. 41.

[13] O. Sporns, *Networks of the Brain*: MIT press, 2010.

[14] S. Samarasinghe, *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*: CRC Press, 2016.

[15] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences,* vol. 378, pp. 484-497, 2017.

[16] S. Goyal and G. K. Goyal, "Heuristic machine learning feedforward algorithm for predicting shelf life of processed cheese," *International Journal of Basic and Applied Sciences,* vol. 1, pp. 458-467, 2012.

[17] S. Goyal and G. K. Goyal, "Soft computing single hidden layer models for shelf life prediction of burfi," *Russian Journal of Agricultural and Socio-Economic Sciences,* vol. 5, 2012.

[18] G.-z. Quan, Z.-y. Zhan, T. Wang, and Y.-f. Xia, "Modeling the Hot Tensile Flow Behaviors at Ultra-High-Strength Steel and Construction of Three-Dimensional Continuous Interaction Space for Forming Parameters," *High Temperature Materials and Processes,* vol. 36, pp. 29-43, 2017.

[19] C. MacLeod, "The synthesis of artificial neural networks using single string evolutionary techniques," 1999.

[20] J. Nagi and M. S. K. AHMED, "Pattern Recognition Of Simple Shapes In A Matlab/Simulink Environment: Design And Development Of An Efficient High-Speed Face Recognition System," *A Thesis Electrical And Electronics Engineering. University Tenaga Nasional,* 2007.

[21] P. Auer, H. Burgsteiner, and W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," *Neural Networks,* vol. 21, pp. 786-795, 2008.

[22] Y.-C. Hu, "Tolerance rough sets for pattern classification using multiple grey single-layer perceptrons," *Neurocomputing,* vol. 179, pp. 144-151, 2016.

[23] Q. V. Le, "A Tutorial on Deep Learning Part 1: Nonlinear Classifiers and The Backpropagation Algorithm," ed, 2015.

[24] B. Choubin, S. Khalighi-Sigaroodi, A. Malekian, and Ö. Kişi, "Multiple linear regression, multi-layer perceptron network and adaptive neuro-fuzzy inference system for forecasting precipitation based on large-scale climate signals," *Hydrological Sciences Journal,* vol. 61, pp. 1001-1009, 2016.

[25] S. P. Fard and Z. Zainuddin, "The universal approximation capabilities of double 2\ pi-periodic approximate identity neural networks," *Soft Computing,* vol. 19, pp. 2883-2890, 2015.

[26] H. H. Bhadeshia, "Neural networks in materials science," *ISIJ international,* vol. 39, pp. 966-979, 1999.

[27] H. Schütze, D. A. Hull, and J. O. Pedersen, "A comparison of classifiers and document representations for the routing problem," in *Proceedings of the 18th*

annual international ACM SIGIR conference on Research and development in information retrieval, 1995, pp. 229-237.

[28] M. M. Saggaf, M. N. Toksöz, and H. M. Mustafa, "Estimation of reservoir properties from seismic data by smooth neural networks," *Geophysics,* vol. 68, pp. 1969-1983, 2003.

[29] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *Information Theory, IEEE Transactions on,* vol. 44, pp. 525-536, 1998.

[30] H. Xiao and X. Zhu, "Margin-Based Feed-Forward Neural Network Classifiers," *arXiv preprint arXiv:1506.03626,* 2015.

[31] B. B. Benuwa, Y. Z. Zhan, B. Ghansah, D. K. Wornyo, and F. Banaseka Kataka, "A Review of Deep Machine Learning," in *International Journal of Engineering Research in Africa*, 2016, pp. 124-136.

[32] P. Vishwanath and V. Viswanatha, "FACE CLASSIFICATION USING WIDROW-HOFF LEARNING PARALLEL LINEAR COLLABORATIVE DISCRIMINANT REGRESSION (WH-PLCDRC)," *Journal of Theoretical and Applied Information Technology,* vol. 89, p. 362, 2016.

[33] J. Wang, W. Wu, Z. Li, and L. Li, "Convergence of gradient method for double parallel feedforward neural network," *Int J Numer Anal Model,* vol. 8, pp. 484-495, 2011.

[34] L. S. D. G. Z. Shisheng, "Aeroengine Lubricating Oil Metal Elements Concentration Prediction Based on Double Parallel Process Neural Network [J]," *Lubrication Engineering,* vol. 5, p. 010, 2006.

[35] X. MENG, G.-b. DING, and L. TANG, "Calculation for the Exhaust Enthalpy of a Steam Turbine Based on Parallel Connection Feed-forward Network," *Turbine Technology,* vol. 1, p. 004, 2006.

[36] G. Huang and R. He, "Analyzing water diversion demand for irrigation areas at lower reach of yellow river with BP neural network techniques," *J. Irriga. Drain,* vol. 19, pp. 20-23, 2000.

[37] M. He, "Double Parallel Feedforward Neural Networks with Application to Simulation Study of Flight Fault Inspection," *Acta. Aerona. ET. Astrona. Sinica,* vol. 15, pp. 877-881, 1994.

[38] S. Haykin and R. Lippmann, "Neural Networks, A Comprehensive Foundation," *International Journal of Neural Systems,* vol. 5, pp. 363-364, 1994.

[39] C. G. Looney, *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*: Oxford University Press, Inc., 1997.

[40] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks,* vol. 6, pp. 861-867, 1993.

[41] Y. Liang, D. Feng, H. P. Lee, S. P. Lim, and K. Lee, "Successive approximation training algorithm for feedforward neural networks," *Neurocomputing,* vol. 42, pp. 311-322, 2002.

[42] Y. Liang, W. Lin, H. Lee, S. Lim, K. Lee, and H. Sun, "Proper orthogonal decomposition and its applications–part II: Model reduction for MEMS dynamical analysis," *Journal of Sound and Vibration,* vol. 256, pp. 515-532, 2002.

[43] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks,* vol. 4, pp. 251-257, 1991.

[44] S.-s. Zhong and G. Ding, "Research on double parallel feedforward process neural networks and its application," *Control and Decision,* vol. 20, p. 764, 2005.

[45] D. Wei, "Alternate Iterative Algorithm of Double Parallel Artifical Neural Network and Its Application," *MINIMICRO SYSTEMS-SHENYANG-,* vol. 17, pp. 65-68, 1996.

[46] M. He, "Error Analysis of Double Parallel Feedforward Neural Networks," *JOURNAL-NORTHWESTERN POLYTECHNICAL UNIVERSITY,* vol. 15, pp. 125-130, 1997.

[47] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems,* vol. 4, pp. 950-957, 1995.

[48] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *Neural Networks, IEEE Transactions on,* vol. 16, pp. 57-67, 2005.

[49] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning,* vol. 2, pp. 1-127, 2009.

[50] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160-167.

[51] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 3642-3649.

[52] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification Supplementary Online Material."

[53] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation,* vol. 18, pp. 1527-1554, 2006.

[54] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems,* vol. 19, p. 153, 2007.

[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249-256.

[56] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *The Journal of Machine Learning Research,* vol. 10, pp. 1-40, 2009.

[57] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik, "Inference with the universum," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 1009-1016.

[58] B. T. C. G. D. Roller, "Max-margin Markov networks," *Advances in neural information processing systems,* vol. 16, p. 25, 2004.

[59] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin classification of data with absent features," *The Journal of Machine Learning Research,* vol. 9, pp. 1-21, 2008.

[60] R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin based feature selection-theory and algorithms," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 43.

[61] B. Li, M. Chi, J. Fan, and X. Xue, "Support cluster machine," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 505-512.

[62] B. Ghansah, S. Wu, and N. Ghansah, "Rankboost-Based Result Merging," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, 2015, pp. 907-914.

[63] T. N. Huynh and R. J. Mooney, "Online Max-Margin Weight Learning for Markov Logic Networks," in *SDM*, 2011, pp. 642-651.

[64] M. Hoai and F. De la Torre, "Max-margin early event detectors," *International Journal of Computer Vision,* vol. 107, pp. 191-202, 2014.

[65] C. Li, Q. Liu, W. Dong, F. Wei, X. Zhang, and L. Yang, "Max-Margin-Based Discriminative Feature Learning," *IEEE transactions on neural networks and learning systems*, vol. 27, pp. 2768-2775, 2016.

[66] S. A. Ali, M. Andleeb, and R. Asif, "Performance Evaluation of Loss Functions for Margin Based Robust Speech Recognition," *Performance Evaluation,* vol. 7, 2016.

[67] A. Laudani, G. M. Lozito, F. R. Fulginei, and A. Salvini, "On training efficiency and computational costs of a feed forward neural network: a review," *Computational intelligence and neuroscience,* vol. 2015, p. 83, 2015.

[68] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems,* vol. 2, pp. 303-314, 1989.

[69] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks,* vol. 2, pp. 359-366, 1989.

[70] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks*, vol. 2, pp. 183-192, 1989.

[71] J. L. Castro, C. J. Mantas, and J. Benıtez, "Neural networks with a continuous squashing function in the output are universal approximators," *Neural Networks,* vol. 13, pp. 561-563, 2000.

[72] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*: GMD-Forschungszentrum Informationstechnik, 2002.

[73] H. Jaeger, "Echo state network. Scholarpedia 2 (9): 2330," ed, 2007.

[74] T. Lin, B. G. Horne, P. Tiňo, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *Neural Networks, IEEE Transactions on,* vol. 7, pp. 1329-1338, 1996.

[75] A. Rodan and P. Tiňo, "Minimum complexity echo state network," *Neural Networks, IEEE Transactions on,* vol. 22, pp. 131-144, 2011.

[76] D. Li, M. Han, and J. Wang, "Chaotic time series prediction based on a novel robust echo state network," *Neural Networks and Learning Systems, IEEE Transactions on,* vol. 23, pp. 787-799, 2012.

[77] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural networks*, vol. 3, pp. 551-560, 1990.

[78] E. Soria-Olivas, J. D. Martín-Guerrero, G. Camps-Valls, A. J. Serrano-López, J. Calpe-Maravilla, and L. Gómez-Chova, "A low-complexity fuzzy activation function for artificial neural networks," *IEEE Transactions on Neural Networks,* vol. 14, pp. 1576-1579, 2003.

[79] A. L. Braga, C. H. Llanos, D. Göhringer, J. Obie, J. Becker, and M. Hübner, "Performance, accuracy, power consumption and resource utilization analysis for hardware/software realized Artificial Neural Networks," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, 2010, pp. 1629-1636.

[80] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Potentials, IEEE,* vol. 13, pp. 27-31, 1994.