ISSN No. 0976-5697

Volume 5, No. 2, March 2014 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Exemplifying Workflow Sequencing and Analysis in Artificial Neural Networks

Ramachandra Rao Kurada
Asst. Prof., Department of Computer Applications
Shri Vishnu Engineering College for Women
Vishnupur - Bhimavaram
ramachandrarao.kurada@gmail.com

Dr. Karteeka Pavan Kanadam
Professor, Department of IT
RVR & JC College of Engineering
Chowdavaram- Guntur
kanadamkarteeka@gmail.com

Abstract: Artificial Neural networks have seen an flare-up of attention over the most recent years and are being productively functional across an astonishing variety of problem domains, varied as science, finance, medicine, engineering, physics and biology. The exhilaration track from the fact that these networks are cracked to model with the competence of the human brain. From a statistical viewpoint artificial neural networks are fascinating because of their prospective use in prediction, regression and classification tribulations. This paper advocates the significance of Artificial Neural Networks by highlighting its advancements, trends and challenges. In addition, this study aims to magnetize research appetizers with a road map towards application solving and psychiatry in a methodical approach.

Keywords: Artificial Neural Networks; Supervised learning; Unsupervised learning; Workflow Sequencing; Real-time datasets

I. INTRODUCTION

Recent developments in science and technology have enabled the growth and availability of raw data to occur at an explosive rate. This has created an immense opportunity for knowledge discovery and data engineering research to play an essential role in a wide range of applications. The study of Artificial neural networks (ANNs) [1] has aroused great interest as they are universal function approximations capable of mapping any linear or nonlinear function. ANNs flexibility in function approximation make them very useful in tasks involving pattern classification, estimating continuous variables, forecasting, and business surveys etc [2], [3].

Artificial intelligence, neural computing, and pattern recognition share a common knowledge base comprising of multiple disciplines. Contemporary neurocomputing takes its models from the biological system. Human brain is the basic motivation in the endeavor to building intelligent machine in the field of artificial intelligence. The idea of creating a network of neurons got a boost when McCulloch and Pitts presented their model of the artificial neuron laying the foundations [4]. Much work was done in the field to a point where simulations of the net could be performed on computers.

ANNs models capitalize on properties of biological gene networks that other kinds of models do not. ANNs naturally take advantage of patterns of absence, as well as presence, of factor binding associated with specific expression output. ANNs are easily subjected to in silico ‘‘mutation’’ to uncover biological redundancies, and can use the full range of factor binding values [5]. ANNs are structural computational models with a long history in pattern recognition. A general reason for thinking ANNs could be effective for this task is that they have some natural similarities with transcription networks, including the ability to create nonlinear sparse interactions between transcriptional regulators and target genes [6].

ANNs have been most famously used in machine learning as ‘‘black boxes’’ to perform classification tasks, in which the goal is to build a network based on a training dataset that will subsequently be used to perform similar classifications on new data of similar structure. In these classical ANNs applications, the weights within the network are of no particular interest, as long as the trained network performs the desired classification task successfully when extrapolating to new data [7].

The number of types of ANNs and their uses is very high. Since the first neural model by McCulloch and Pitts there have been developed hundreds of different models considered as ANNs. The differences in them might be the functions, the accepted values, the topology, the learning algorithms, etc. In addition, there are many hybrid models where each neuron has more properties for engineering purposes, such as pattern recognition, forecasting, and data compression [8]. ANNs posses a number of properties for modeling processes or systems with universal function approximation capability, learning from experimental data, tolerance to noisy or missing data, and good generalization capability [9].

The backpropagation algorithm proposed by Rumelhart and McClelland [10] uses a layered feed-forward ANNs. This means that the artificial neurons are organized in layers, and send their signals forward, and then the errors are propagated backwards. The network receives inputs by neurons in the input layer, and the output of the network is given by the neurons on an output layer. There may be one or more intermediate hidden layers. The back propagation algorithm uses supervised learning, which means the algorithm with examples of the inputs and outputs we want the network to compute, and then the error (difference between actual and expected results) is calculated. The idea of the back propagation algorithm is to reduce this error, until the ANN learns the training data. The training begins with random weights, and the goal is to adjust them so that the error will be minimal.

The overall organization of the paper is as follows. Section I Introduction, presents the fundamental of ANNs, issues and outlines the general stages, task decomposition strategies etc. Section II Workflow analysis inducts the roadmap to solve the practical problems with a systematic approach in seven stages. This section provokes a quick reference to major attempts in theoretical issues of learning and generalization in classification of ANNs, transfer, activation, output functions etc. Section III Experimental analysis is worthwhile with comparisons, impressions on diverse applications and implementations of ANNs models over real-time datasets. Finally, Section IV concludes the paper with general recommendations for future designs and learning performances.

II. WORKFLOW ANALYSIS AND SEQUENCING

A. Data Collection:

Data is specific to its application domain and is preprocessing. Neural network training can be more efficient if certain preprocessing steps are applied on the network inputs and targets. Generally, the normalization step is applied to both the input vectors, the target vectors in the data set, and the network output always falls into a normalized range. The network output can then be reverse transformed back into the units of the original target data when the network is put to use in the field. Real time dataset are available at the UCI Machine Learning Repository http://mlearn.ics.uci.edu/MLRepository.html. These datasets are originated from the StatLib library, which is maintained at Carnegie Mellon University [11].

The most Popular datasets considered from UCI Machine Learning Repository are cancer, iris and glass datasets. The extracted data is preprocessed and divided into subsets before it is supplied into network as input. Since it is generally difficult to incorporate prior knowledge into a neural network, therefore the network can only be as accurate as the data that are used to train the network. It is important that the data cover the range of inputs for which the network will be used [12]. Multilayer networks can be trained to generalize well within the range of inputs for which they have been trained. However, they do not have the ability to accurately the extrapolate beyond this range, so it is important that the training data span the full range of the input space [13].

When training multilayer networks, the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to rise. The network weights and biases are saved at the minimum of the validation set error [14].

B. Network Creation:

After the data has been preprocessed, the next step is to create a network object and train the network. The ANNs can be created and trained with supervised and unsupervised learning methods [15].

a. Feed forward backpropagation (FB) Model:

FB artificial intelligence model [16] consists of input, hidden and output layers. Backpropagation learning algorithm is used for learning these networks. During training FB network, calculations were carried out from input layer of network toward output layer, and error values were then propagated to prior layers. Feedforward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range –1 to +1. Thus the output layer in order to produce values between 0 and 1uses a sigmoid transfer function [17].

b. Cascade forward (CF) Model:

CF models are similar to feed-forward networks [18], but include a weight connection from the input to each layer and from each layer to the successive layers. While two-layer feedforward networks can potentially learn virtually any input-output relationship, feed-forward networks with more layers might learn complex relationships more quickly. CF artificial intelligence model is similar to FB model in using the backpropagation algorithm for weights updating, but the main symptom of this network is that each layer of neurons related to all previous layer of neurons. Tan-sigmoid transfer function [19], log - sigmoid transfer function [20] and pure linear threshold functions were used to reach the optimized status [21].

c. Competitive Network Model"

A competitive learning network comprises the feed forward excitatory network and the lateral inhibitory networks [22]. The feedforward network usually implements an excitatory Hebbian learning rule [23]. It consist of an input cell persistently participates in firing an output cell, the input cell's influence firing that output cell is increased. The lateral competitive network is inhibitory in nature. The network serves the important role of selecting the winner, often via a competitive learning process, highlighting the "winner-take-all" schema. In a winner-take-all circuit, the output unit receiving the largest input is assigned a full value i.e. 1, whereas all other units are suppressed to a 0 value.

d. ELMan Neural Network (ENN) Model:

ENN [24] is one type of the partial recurrent neural networks, which consists of a two-layer back propagation network with an additional feedback connection from the output of the hidden layer to its input. The advantage of this feedback path is that it allows the ENN to recognize and generate temporal patterns and spatial patterns. This means that after training, interrelations between the current input and internal states are processed to produce the output and to represent the relevant past information in the internal states. As a result, the ENN has been widely used in various fields which includes classification, prediction and dynamic system identification, etc. The overall structure of ENN is shown in Fig. 1.

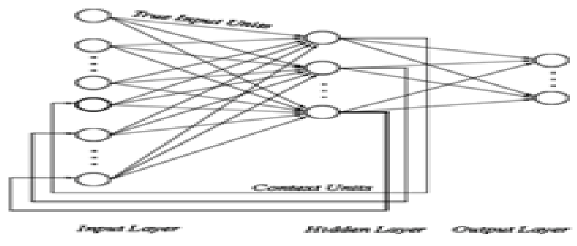

Figure 1. Structure of Elman Neural Network

e. Generalized Regression Neural Network (GRNN):

GRNN [25] is often used for function approximation. It has a radial basis layer and a special linear layer. This GRNN is a one-pass learning algorithm with a parallel structure. Even with sparse data in a multidimensional measurement space, the algorithm provides smooth transitions from one observed value to another. This algorithmic form can be used for any regression problem in which an assumption of linearity is not justified.

This network like other probabilistic neural networks [26] needs only a fraction of the training samples a back propagation neural network. Therefore, the use of a probabilistic neural network is especially advantageous due to its ability to converge to the underlying function of the data with only few training samples available. This makes GRNN a very useful tool to perform predictions and comparisons of system performance in practice.

f. Hopfield Neural Network (HNN) Model:

John Hopfield of the California Institute of Technology proposed the Hopfield model during the early 1980s [27]. The HNN is perhaps the simplest of ANN; it is a fully connected single layer auto associative network. This means it has one single layer, with each neuron connected to every other neuron. Hopfield networks are a special kind of recurrent neural networks that can be used as associative memory.

Associative memory is often addressed through its contents i.e. if a pattern is presented to an associative memory, it returns whether this pattern coincides with a stored pattern. The coincidence need not be perfect, though. An associative memory may also return a stored pattern that is similar to the presented one, so that noisy input can also be recognized [28]. Neurons are pixels and can take the values of -1 or +1. The network has stored a certain number of pixel patterns. During a retrieval phase, the network is started with some initial configuration and the network dynamics evolves towards the stored pattern, which is closest to the initial configuration.

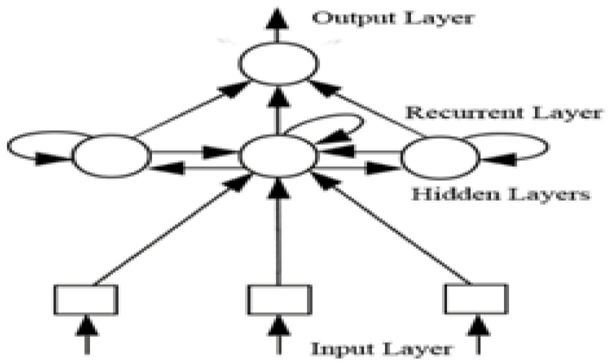

Figure 2. Structure of Layer Recurrent Neural Network

g. Layer Recurrent Neural Network (LRNN) Model:

The fundamental feature of a LRNN is that the network contains at least one feed-back connection, so that activation can flow round in a loop [29]. That enables the networks to do temporal processing and learn sequences. The architectures of recurrent neural networks can take many different forms, but they all share two important common features i.e. to incorporate some form of MLP as a sub-system and to exploit the powerful non-linear mapping capabilities of the MLP with some form of memory [30]. The relevant areas where LRNN seem to be very promising for modeling and simulation includes are neuro identification, neuro control, diagnosis and forecasting, the overall structure of LRNN is shown in Fig. 2.

h. Learning Vector Quantization (LVQ) Network Model:

LVQ is a supervised version of vector quantization that can be used when we have labeled input data [31]. This learning technique uses the class information to reposition the Voronoi vectors slightly, to improve the quality of the classifier decision regions. LVQ has a two stage process i.e. Self-Organizing Map (SOM) followed by LVQ. This model is particularly useful for pattern classification problems. The first step is feature selection i.e. the unsupervised identification of a reasonably small set of features in which the essential information content of the input data is concentrated. The second step is the classification where the feature domains are assigned to individual classes.

LVQ model first has a competitive layer and is followed by a linear layer. The competitive layer learns to classify input vectors in much the same way as the competitive layers of Cluster with SOM. The linear layer transforms the competitive layer's classes into target classifications defined by the user. The classes learned by the competitive layer are referred to as subclasses and the classes of the linear layer as target classes. Both the competitive and linear layers have one neuron per (sub or target) class.

i. Probabilistic Neural Networks (PNN) Model:

PNN are used for classification problems [32]. When an input is presented, the first layer computes distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a compete transfer function on the output of the second layer picks the maximum of these probabilities, and produces 1 for that class and a 0 for the other classes.

j. Radial Basis Function (RBF):

RBF Networks [33] take a slightly different approach of MLP. RBF is a two-layer feed-forward networks. The hidden node is implemented with a set of radial basis functions (e.g. Gaussian functions), the output nodes are implemented as a linear summation functions as in an MLP. The network training is divided into two stages i.e. the weights from the input to hidden layer are first determined, and then the weights from the hidden to output layer are determined.

k. Self Organized Map (SOM) Model:

SOMs [34] are used both to cluster data and to reduce the dimensionality of data. They are inspired by the sensory and motor mappings in the mammal brain, which also appear to automatically organizing information topologically. The principal goal of an SOM is to transform an incoming signal pattern of arbitrary dimension into a one or two-dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion.

Kohonon's SOMs are a type of unsupervised learning and it is used to discover some underlying structure of the data.

Kohonen's SOM is called a topology-preserving map because there is a topological structure imposed on the nodes in the network [35]. A topological map is simply a mapping that preserves neighborhood relations. Therefore SOM is setup by placing neurons at the nodes of a one or two dimensional lattice. Higher dimensional maps are also possible, but not so common. The neurons become selectively tuned to various input patterns (stimuli) or classes of input patterns during the course of the competitive learning. The locations of the neurons so tuned (i.e. the winning neurons) become ordered and a meaningful coordinate system for the input features is created on the lattice. The SOM thus forms the required topographic map of the input patterns.

SOM consists of a competitive layer which can classify a dataset of vectors with any number of dimensions into as many classes as the layer has neurons. The neurons are arranged in a 2D topology, which allows the layer to form a representation of the distribution and a two-dimensional approximation of the topology of the dataset. The network is trained with the SOM batch algorithm (trainbu, learnsomb). SOM is a vector quantization method, which places the prototype vectors on a regular low-dimensional grid in an ordered fashion. This makes the SOM a powerful visualization tool.

The feed forward structure of Kohenon SOM is illustrated in Fig. 3. The SOM has a feed-forward structure with a single computational layer arranged in rows and columns. Each neuron is fully connected to all the source nodes in the input layer. A clear understanding from Fig. 3 was a one dimensional map will just have a single row (or a single column) in the computational layer.

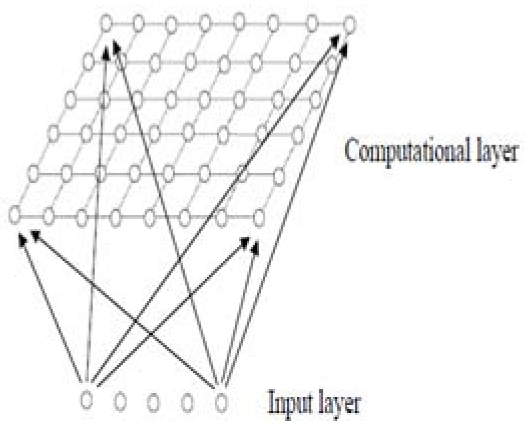

Figure 3. Feed-forward structure of Kohenon SOM network

The self-organization process involves four major stages:

a. ***Initialization:*** All the connection weights are initialized with small random values.
b. ***Competition:*** For each input pattern, the neurons compute their respective values of a discriminated function that provides the basis for competition. The particular neuron with the smallest value of the discriminated functions is declared the winner.
c. ***Cooperation:*** The winning neuron determines the spatial location of a topological neighborhood of excited neurons, thereby providing the basis for cooperation among neighboring neurons.
d. ***Adaptation:*** The excited neurons decrease their individual values of the discriminated function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

There are two identifiable phases of this adaptive process:

a) ***Ordering or self-organizing phase***– here the topological ordering of the weight vectors takes place. Typically, this will take as many as 1000 iterations of the SOM algorithm, and careful consideration needs to be given to the choice of neighborhood and learning rate parameters.
b) ***Convergence phase–*** during which the feature map is fine tuned and comes to provide an accurate statistical quantification of the input space. Typically, the number of iterations in this phase will be at least 500 times the number of neurons in the network, and again the parameters must be chosen carefully.

The stages of the SOM algorithm can be summarized as follows:

i. ***Initialization –*** Choose random values for the initial weight vectors w_j
ii. ***Sampling –*** Draw a sample training input vector x from the input space.
iii. ***Matching –*** Find the winning neuron I(x) with weight vector closest to input vector.
iv. ***Updating –*** Apply the weight update equation $\Delta W_{ij}=\eta(t), I(x)(t)(xi-W_{ji})$
v. ***Continuation –*** keep returning to step 2 until the feature map stops changing.

C. Network Design:

The configuration of network is done by selecting the number of hidden layers, number of neurons in each hidden layers and transfer functions. In backpropagation it is important to be able to calculate the derivatives of any transfer functions used. Feedforward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range -1 to +1. Each of the transfer functions logsig, tansig, and purelin, calculate their own derivative. The function logsig generates outputs between zero and one as the neuron's net input goes from negative to positive infinity. If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value.

D. Determinationof Weights and Biases:

Once the network is chosen and configured with input, hidden and output layers, the training function, adaptation learning function [36] and performance learning functions [37] are also to be constituted. Few networks training function available in matlab software are trainbfg which updates weight and bias values according to the BFGS quasi-Newton method [38], traingdm, traingd functions that updates weight and bias values according to gradient descent with momentum, trainlm function which updates weight and bias values according to Levenberg-Marquardt optimization [39]. It is often the fastest backpropagation algorithm and is highly recommended as a first-choice supervised algorithm.

The other training functions like traincgb updates weight and bias values according to the conjugate gradient backpropagation with Powell-Beale [40] restarts, traincgf updates weight and bias values according to conjugate gradient backpropagation. The adaptation learning functions like learngd, learngdm uses gradient descent with momentum weight and bias learning function. The Network performance function like MSE is used to measure the network's performance according to the mean of squared errors. The Mean squared error with regularization performance function, msereg is used to measures network performance as the weight sum of two factors i.e. the mean squared error and the mean squared weight and bias values.

E. *Training the Neural Network:*

When the network weights and biases are initialized, the network is ready for training. The multilayer feed forward network can be trained for function approximation (nonlinear regression) or pattern recognition. The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance.

There are two different ways in which training can be implemented: incremental mode and batch mode [41]. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. The fastest training function is generally trainlm, and it is the default training function for feedforward network. The quasi-Newton method, trainbfg, is also relatively quite faster method as a training function. In addition, trainlm performs better on function fitting (nonlinear regression) problems than on pattern recognition problems. When training large networks, and when training pattern recognition networks, trainscg and trainrp are good choices. Their memory requirements are relatively small, and yet they are much faster than standard gradient descent. Transfer functions calculate a layer's output from its net input. Hyperbolic tangent sigmoid transfer function tansig is one amongst it. This function calculate a layer's output from its net input and it is a good tradeoff for neural networks, where speed is important and the exact shape of the transfer function is not. Once the training process is completed, the network must be able to classify or predict from new inputs.

The network will determine the entire coefficient by back-propagation of errors, which will try to maximize the sum of squares of the difference (errors) between the expected and the actual computed output. This process usually takes hundreds or thousands of iterations. The rate of convergence is faster in earlier iterations and becomes slower as the iteration number increases. If a reasonable degree of convergence is considered, the network is trained and can be imposed on real world applications, domains etc.

F. *Network Validation:*

One of the major advantages of ANNs is their ability to generalize. This means that a trained net could classify data from the same class as the learning data that it has never seen before. In real world applications, developers normally have only a small part of all possible patterns for the generation of a neural net. To reach the best generalization, the dataset is split into three parts:

i. ***Training Set:*** It is used to train a neural net. The error of this dataset is minimized during training.
ii. ***Validation set:*** It is used to determine the performance of a neural network on patterns that are not trained during learning.
iii. ***Testing set:*** It is used for finally checking the overall performance of a neural net.

The learning stops when it produces a minimum of the validation set error. At this point the ANN generalizes the best. When learning is not stopped, overtraining occurs and the performance of the net overall data decreases, despite the fact that the error on the training data still gets smaller. After finishing the learning phase, the model should be finally checked with the third data set, the test set.

The evaluation and validation of an ANNs prediction model are based upon one or more selected error metrics [42]. Generally, ANNs models, a function approximation task that will use a continuous error metric such as mean absolute error (MAE), mean squared error (MSE) or root mean squared error (RMSE). These errors are summed over the validation set of inputs and outputs, and then normalized by the size of the validation set.

G. *Network Exploitation:*

There are numerous fields where neural system is being used since they are good when dealing with abstract problems, like those based on features and patterns. ANNs are actively being used for applications as bankruptcy prediction, predicting costs, forecast revenue, processing documents and more. The other major benefits of ANN are they are inherently multiprocessor-friendly architecture, and have ability to do many things at once and provide vital information for powerful decision-making [43]. Depending on the nature of the application and the strength of the internal data patterns, generally the network is expected to train quite well. This applies to problems where the relationships may be quite dynamic or non-linear. ANNs provide an analytical alternative to conventional techniques, which are often limited by strict assumptions of normality, linearity, variable independence etc. Because an ANN can capture many kinds of relationships it allows the user to quickly and relatively easily model phenomena which otherwise may have been very difficult or impossible to explain otherwise. Neural networks are universal approximations, and they work best if the system you are using them to model has a high tolerance to error.

III. EXPERIMENAL ANALYSIS

Once a network model is customized and tailored to a particular application, that network is ready to be trained. The model is initialized with a weight randomly and learns the content of the dataset. The two approaches to train a model are the supervised and unsupervised.

Table I: Experimental Analysis of Feed Forward Back Propagation Model

Dataset	Epoch	Time (sec)	Performance	Gradient	MU	Validation Checks	MSE	Regression
Iris	13	0.28	2.28e-09	7.92e-06	1.00e-09	6	9.87195e-3	9.77540e-1
Cancer	10	0.018	0.000770	0.0100	0.0182	6	1.50737e-2	9.69921e-1
Glass	13	0.01	0.0161	0.00811	0.100	5	7.89894e-0	9.52435e-1
Body fat	10	0.01	7.78	11.4	10	5	12.61533e-0	9.07788e-1
Building Energy	35	0.11	0.00237	0.000386	0.0100	6	2.31464e-3	9.25453e-1
Housing	6	0.01	3.41	8.78	10.0	6	3.34751e-2	9.31548e-1

Supervised training involves a mechanism of providing the network with the desired output either by manually grading the network's performance or by providing the desired outputs with the inputs. In the Unsupervised training the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full-blown sense of being truly self-learning, it is still just a shining promise that is not fully understood.

A. *Dataset Description:*

a. ***house_dataset:*** It estimates the median value of owner occupied homes in Boston suburbs given 13 neighborhood attributes. An estimator can be found by Fitting the inputs and targets. This data set has 506 samples with 13 attributes each. The expected output is a sample with median values of owner-occupied homes in $1000's.

b. ***abalone_dataset:*** It estimates the number of rings of an abalone shell with eight measurements. An estimator can be found by Fitting the inputs and targets. The data set consists of 4177 samples with 8 attributes per each sample.

c. ***bodyfat_dataset:*** This dataset can be used to train a neural network to estimate the bodyfat of someone from 13 measurements per sample out of 252.

d. ***building_dataset:*** This dataset can be used to train a neural network to estimate the energy use of a building from time and weather conditions. It is defined with 14 attributes from 4208 samples.

B. *Evaluation Measures:*

a. ***Mean Squared Error (MSE):*** MSE is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

b. ***Regression (R):*** The Regression values R measure the correlation between outputs and targets. R value is of 1 means a close relationship, 0 a random relationship. Regression procedures are like correlation because they are concerned with relationships among variables. Correlation analyses serve as the part of the building block for regression procedures.

C. *Supervised LearningTechniques:*

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs repeatedly as the weights are continually tweaked. The set of data, which enables the training, is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

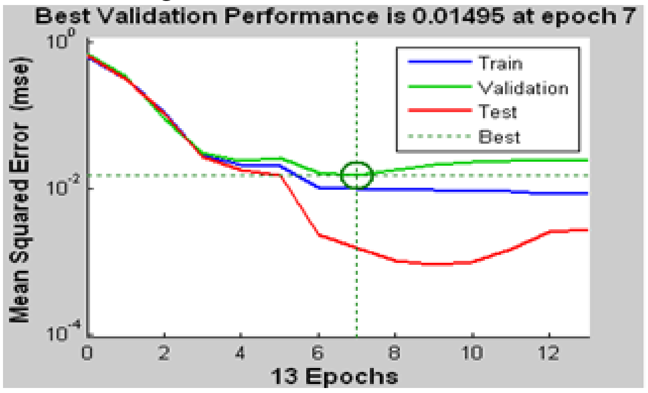

Figure 4. Best Validation Performance on IRIS Dataset

The outcomes of experimental analysis on FBNN model when applied over the real-time datasets are shown in Table1. The MSE is relatively low for the cancer, building energy, housing datasets when compared with the other datasets like iris, glass, and body fat. The Regression coefficient is almost near to one to all the comparing dataset, which means a positive sign that training is almost approximate. Building energy dataset consumes more number of iterations due its huge number of samples in training set. The glass and bodyfat datasets consume less number of CPUs training time and it reflects by consuming less number of validation checks. The overall performance of FB model is optimum in Building dataset and reflects the same in MU. The best validation performance of iris dataset at epoch 7 is shown as Fig. 4. All the iterations are assumed in x-axis and are corresponded with MSE in y-axis. The three divisions of dataset training, validation and testing sets are also depicted in the Fig 4.

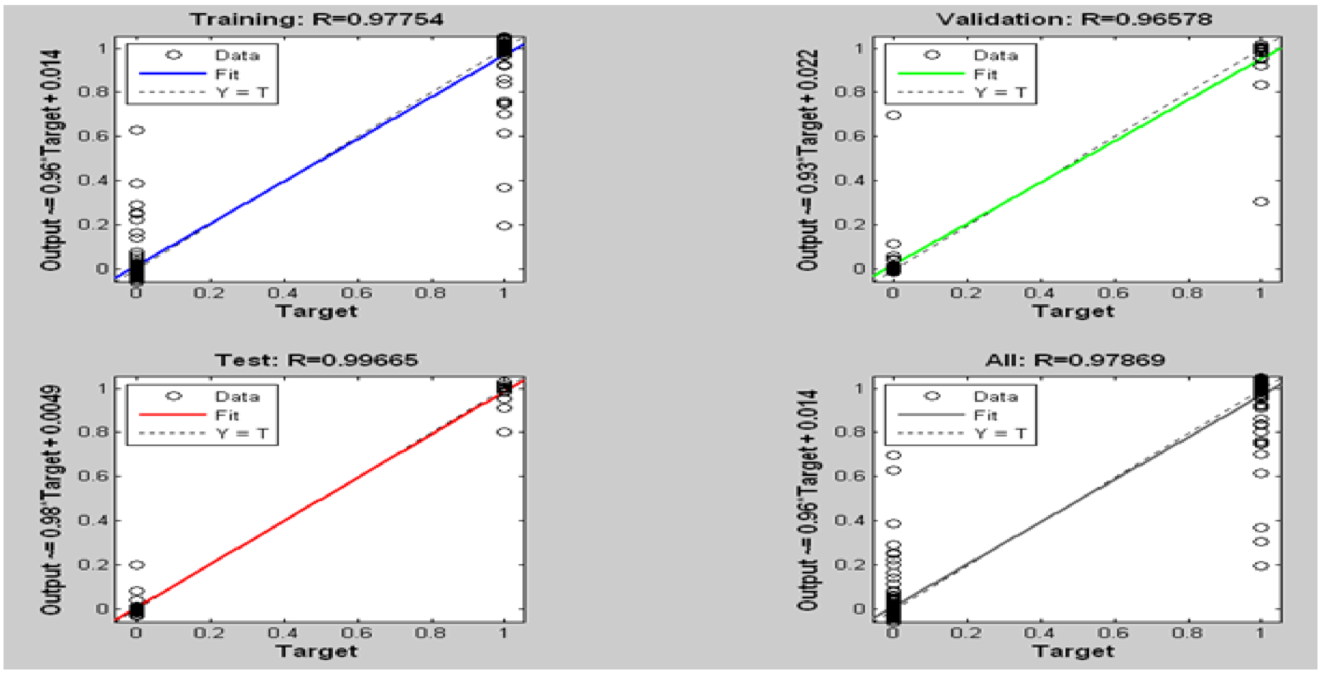

Figure 5. Regression values of Training, Validation, Testing and overall sets on iris dataset

Table 2. Experimental analysis of Cascade forward Model

Dataset	Epoch	Time (sec)	Performance	Gradient	MU	Validations
Iris	15	0.01	0.0114	0.0279	0.0100	6
Cancer	11	0.02	1.15e-07	2.43e-06	1.00e-08	5
Glass	16	0.01	0.00440	0.000725	0.00100	6
Housing	14	0.13	3.93	32.7	10	6
Body fat	11	0.01	7.09	9.89	10	6
Abalone	5	0.10	90.1	4.07e-08	0.00100	0

Table 3. Experimental analysis of Elman Neural Network Model

Dataset	Epoch	Time (sec)	Performance	Gradient	MU	Validations
Iris	11	0	0.126	0.0107	0.00100	6
Cancer	13	0	0.00870	0.0509	0.0100	6
Glass	17	0	0.00333	1.60e-05	1.00e-07	6

An important observation from Fig. 4 was the best validation performance value 0.01495 is clearly visible at iteration7. Similarly, the Regression value showing the close and random relationship on iris data set in training, testing, validation and overall data is shown as Fig. 5.

Table 2 projects the experimental outcome over the datasets using the CF model. The CF model has tremendous training performance on iris, glass and cancer datasets, but the performance on housing and abalone dataset is high due to more number of samples, but important observation was the number of epoch and validation checks are more or less same to all the dataset. This is again justified by the values in Gradient and Mu column. The gradient and Mu values are low to iris, glass dataset and high to housing, abalone datasets.

Table 3 projects the experimental outcome over the datasets using the Elman Neural Network model. Since the experiments were carried out on relatively small size datasets they do not consume much CPU time, the performance of glass dataset is better than the comparing datasets even though it consumes more number of iterations and gradient value.

Table IV. Experimental analysis of Layer Recurrent Neural Network Model

Dataset	Epoch	Time (sec)	Performance	Gradient	MU	Validations
Iris	14	0.06	0.00609	0.00684	0.100	6
Cancer	10	0.07	0.00229	0.00262	0.0010	6
Glass	9	0.04	0.00140	0.0309	0.0100	6

Table V. Experimental analysis of Self-Organized Maps

Dataset	Epoch	SOM Size	Time (Sec)	No. of Samples	No. of Misclassifications	Classes	SSE	MAE	RMSE	RAE	RASE
Iris	200	10	0.01	150	4	3	7.81	0.0327	0.1291	1.35	0.37
Cancer	200	10	0.03	699	19	2	14.51	0.2911	0.4215	9.01	8.42
Glass	200	10	0.01	214	21	2	8.20	0.1114	0.2627	2.59	0.95
Body fat	200	10	0.01	252	12	2	9.22	0.0982	0.2144	3.25	1.46
Abalone	200	10	0.17	4177		2					

Table 4 projects the experimental outcome over the datasets using the Layer Recurrent Neural Network model. The datasets iris and cancer consumes more number of iterations when compared with glass dataset and hence the same is justified in the total time put away by the CPU in executing the LRNN model. The same was justified by the performance indicators and gradient values that glass dataset hold the optimum values in these columns.

D. Unsupervised LearningTechniques:

ANNs that attempt unsupervised learning have no target outputs. The system itself must then decide what features it will use to group the input data. This paper referrers the SOM model for learning patterns in datasets. During the learning process, the units (weight values) of ANNs network are arranged inside a certain range, depending on given input values. The goal is to group similar units close together in certain areas of the value range. Table 5 projects the experimental outcome of SOM over real time datasets. The annotations from Table 5 were SOM categorizes the 150 samples of iris into three classes, experimentally and the same trend prolong to the comparing datasets. Another important footnote was the learning time of CPU is almost minimal to all the datasets, which have less than 700 samples and quite a appreciable time to abalone dataset.

The other quantifiers used in Table 5 to evaluate the cluster quality and model SSE, MAE, RMSE, RAE, RASE. Sum of Squared Error (SSE), is used to measure the differences between each sample observation, and its group's mean. A tremendous fall in SSE is recorded when SOM is imposed on datasets. A general impression on SOM was the misclassifications rate is very low despite the number of CPU cycles and iterations. The Mean Absolute Error (MAE) is used measure the closeness between the actual and predicted sample in the dataset. The error rate raised is very low for all the comparing datasets over the SOM model, this justifies lead to a positive sign that this unsupervised model can used for prediction.

Root Mean Square Error (RMSE) is used to find the accuracy of grouping by finding the differences between values predicted by a model and the values actually observed; also, this measure aggregates the values of residuals. An affirmative symptom was RMSE is always phosphorus towards the lower threshold values in all comparing datasets. Relative Absolute Error (RAE) is the total absolute error made relative to what the error would have been if the prediction simply had been the average of the actual values. This precise model is statistically depicting towards zero in all the comparing datasets. The Relative Squared Error (RSE) takes the total squared error and normalizes it by dividing by the total squared error of the simple predictor. This exaggerates the prediction error was slightly greater than actual than the mean error in all comparing cases.

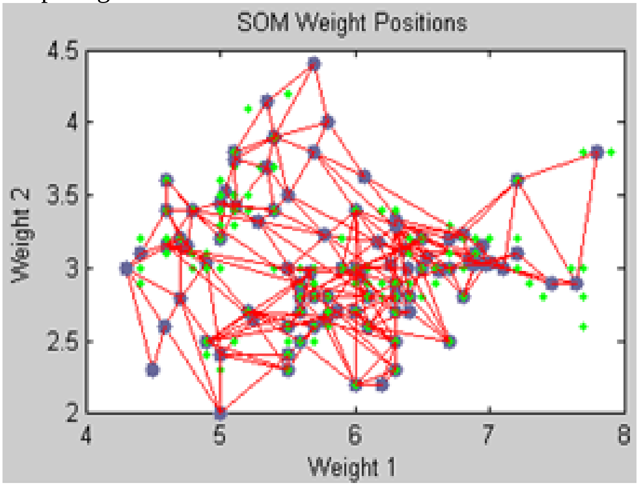

Figure 6. SOM weight positions in iris dataset

Fig. 6 and Fig. 7 illustrates the spread of SOM weight positions between attributes in iris and cancer datasets. The positions and relations are composed with the location of data points and weight vector. The neighbor weight distances of cancer and iris datasets are visualized using SOM in Fig. 8 and Fig. 9. The input space is visualized by the set of neurons, connection between neighbor neurons including the small and large distance between the neurons. Grouping is observed at the light and dark segments in the SOM to indicate that the network has clustered data into groups. Hence, it is justified that SOM learn to classify input vectors according to how they are grouped in the input space. In addition, SOM learn both the distribution and topology of the input vectors they are trained on.

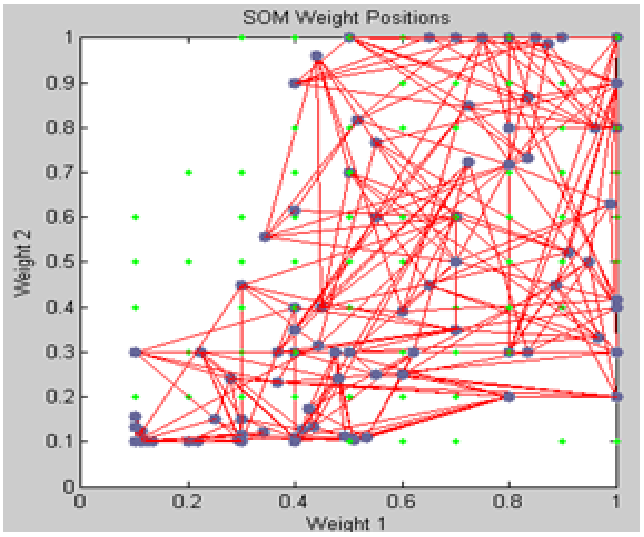

Figure 7. SOM weight positions in cancer dataset

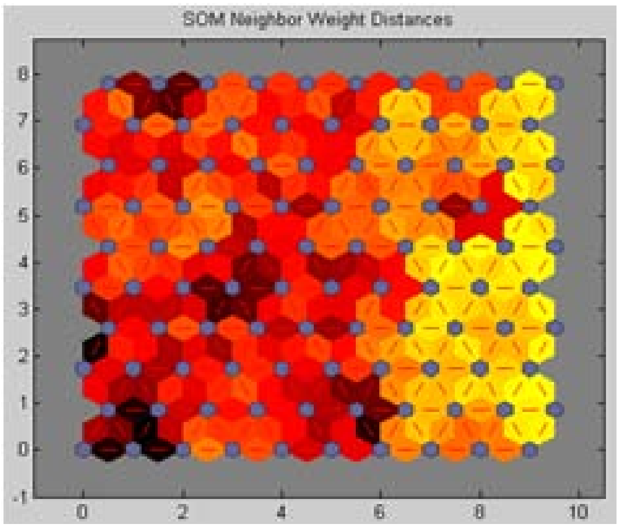

Figure 8. SOM neighbor weight distances in cancer dataset

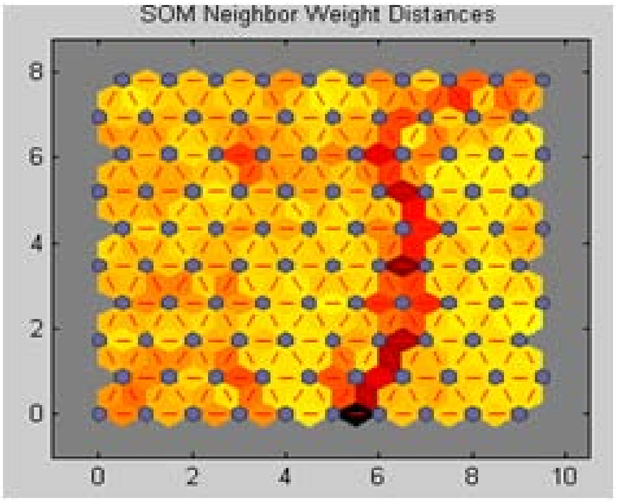

Figure 9. SOM neighbor weight distances in iris dataset

E. Overall Impression:

The ENN model outperforms all the comparing model over the iris, glass and cancer dataset with respect to cpu learning time, but performance wise of the supervised learning models LRNN prevails its supremacist over the datasets. In the body fat dataset the learning time is same both at FB and CFN models but a minor performance variation of value is observed. In housing dataset, the FBNN dominance is observed both at CPUs training time and performance.

IV. CONCLUSION

The computing world has a lot to gain from ANNs. Their ability to learn by example makes them very flexible and powerful. Furthermore, there is no need to devise an algorithm in order to perform a specific task. ANNs are very well suited for real time systems because of their fast response and computational times.

In this paper, a systematic approach to solve critical problem sequentially in knowledge discovery and data engineering fields using supervised and unsupervised models of ANNs was endeavored. Several major assessment techniques in ANNs used to evaluate the fundamental nature of the real time datasets. The results and impressions after using these techniques forecasted the accuracy of ANNs.

These results parody ANNs will serve as a comprehensive resource for existing practitioners and future researches with potential research directions and insights to many opportunities and challenges in ANNs, in their field.

V. REFERENCES

[1] McCulloch, Warren S., and Walter Pitts, “A logical calculus of the ideas immanent in nervous activity.” The Bulletin of Mathematical Biophysics , vol 5, no. 4 , pp. 115-133, 1943.

[2] Hebb, Donald Olding, The organization of behavior: A neuropsychological theory , Psychology Press, 2002.

[3] Zhang, Guoqiang, B. Eddy Patuwo, and Michael Y Hu. “Forecasting with artificial neural networks:: The state of the art.” International journal of forecasting vol. 14, no. 1, pp: 35-62, 1998.

[4] McCulloch, W. and W. Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1943.

[5] Sutton, Richard S., and Andrew G. Barto, Reinforcement learning: An introduction. vol. 1, no. 1. Cambridge: MIT press, 1998.

[6] Wasserman, P. D, Neural computing: Theory and practice. Van Nostrand Reinhold, New York, 1989.

[7] R.A. Chayjan.” Modeling of sesame seed dehydration energy requirements by a soft-computing”. Australian journal of crop science,vol. 4, no.3, pp.180-184,2010

[8] Carpenter, Gail A. "Neural network models for pattern recognition and associative memory." Neural networks vol. 2, no. 4, pp: 243-257, 1989.

[9] Cybenko G, “Approximation by superpositions of a sigmoidal function.”, Mathematical Control, Signal and Systems, vol 2, pp: 303-314, 1989.

[10] Rumelhart, D. and J. McClelland, Parallel Distributed Processing. MIT Press, Cambridge, Mass, 1986.

[11] Murphy,P.M., Aha, D.W, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[12] Funahashi, K, On the approximate realization of continuous mappings by neural networks. Neural Networks vol.2, pp: 183-192, 1989.

[13] Hornik, K., Stinchcombe, M. and White, H, “Multilayer feedforward networks are universal approximations.”, Neural Networks , vol 2. pp: 359-366, 1989.

[14] Golbraikh, Alexander, Min Shen, Zhiyan Xiao, Yun-De Xiao, Kuo-Hsiung Lee, and Alexander Tropsha, “Rational selection of training and test sets for the development of validated QSAR models”, Journal of computer-aided molecular design vol. 17, no. 2, pp: 241-253, 2003

[15] Karayiannis, Nicolaos B., and Glenn Weiqun Mi, “Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques.”, Neural Networks, IEEE Transactions, vol 8, no. 6, pp: 1492-1506, 1997.

[16] Johansson, Erik M., Farid U. Dowla, and Dennis M. Goodman, "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method", International Journal of Neural Systems vol. 2, no. 04, pp: 291-301. 1991

[17] Harrington, Peter de B, "Sigmoid transfer functions in backpropagation neural networks", Analytical Chemistry vol. 65, no. 15. Pp: 2167-2168, 1993

[18] Goyal, Sumit, and Gyandera Kumar Goyal, "Cascade and feedforward backpropagation artificial neural networks models for prediction of sensory quality of instant coffee flavoured sterilized drink", Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition vol. 2, no. 6, pp: 78-82, 2011.

[19] Harrington, Peter de B, "Sigmoid transfer functions in backpropagation neural networks", Analytical Chemistry vol. 65, no. 15, pp: 2167-2168, 1993

[20] Mihalakakou, G., M. Santamouris, and A. Tsangrassoulis. "On the energy consumption in residential buildings", Energy and Buildings, Vol. 34, no. 7, pp: 727-736, 2002.

[21] Goyal, Gyanendra Kumar, and Sumit Goyal, "Cascade Artificial Neural Network Models for Predicting Shelf Life of Processed Cheese", Journal of Advances in Information Technology , vol. 4, no. 2, 2013.

[22] Jayasuriya, Suren, and Zachary P. Kilpatrick, "Effects of Time-Dependent Stimuli in a Competitive Neural Network Model of Perceptual Rivalry", Bulletin of mathematical biology, vol. 74, no. 6, pp: 1396-1426, 2012.

[23] Martinetz, Thomas, "Competitive Hebbian learning rule forms perfectly topology preserving maps.", CANN'93, pp. 427-434. Springer London, 1993.

[24] Mohamad, F. N., M. S. A. Megat Ali, A. H. Jahidin, M. F. Saaid, and M. Z. H. Noor, "Principal component analysis and arrhythmia recognition using Elman neural network", In Control and System Graduate Research Colloquium (ICSGRC), 2013 IEEE 4th, pp. 141-146. 2013.

[25] Ding, Shuo, Xiao Heng Chang, and Qing Hui Wu, "A Study on Approximation Performances of General Regression Neural Network", Applied Mechanics and Materials. Vol. 441 pp: 713-716, 2014.

[26] Ding, Shuo, Xiao Heng Chang, and Qing Hui Wu, "Application of Probabilistic Neural Networks in Fault Diagnosis of Three-Phase Induction Motors", Applied Mechanics and Materials vol. 433, pp: 705-708, 2014.

[27] Hopfield, John J, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the national academy of sciences 79, no. 8, pp:2554-2558, 1982

[28] Samad, Tariq, "Neural network auto-associative memory with two rules for varying the weights", U.S. Patent 5,050,095, issued September 17, 1991.

[29] Maas, Andrew L., Tyler M. O'Neil, Awni Y. Hannun, and Andrew Y. Ng, "Recurrent neural network feature enhancement: The 2nd CHiME challenge" ,In Proceedings The 2nd CHiME Workshop on Machine Listening in Multisource Environments held in conjunction with ICASSP, pp. 79-80. 2013.

[30] Kruse, Rudolf, Christian Borgelt, Frank Klawonn, Christian Moewes, Matthias Steinbrecher, and Pascal Held, "Multi-Layer Perceptrons", In Computational Intelligence, pp. 47-81. Springer London, 2013.

[31] Hammer, Barbara, Daniela Hofmann, Frank-Michael Schleif, and Xibin Zhu, "Learning vector quantization for (dis-) similarities", Neurocomputing (2013).

[32] Specht, Donald F, "Probabilistic neural networks", Neural networks vol. 3, no. 1, pp: 109-118, 1990.

[33] Park, Jooyoung, and Irwin W. Sandberg, "Universal approximation using radial-basis-function networks", Neural computation, vol. 3, no. 2, pp: 246-257, 1991.

[34] Kohonen, Teuvo, "Self-organized formation of topologically correct feature maps", Biological cybernetics, vol. 43, no. 1, pp: 59-69, 1982.

[35] Kohonen, Teuvo, "The self-organizing map", Proceedings of the IEEE 78, no. 9, pp:1464-1480, 1990.

[36] Jacobs, Robert A, "Increased rates of convergence through learning rate adaptation", Neural networks vol. 1, no. 4, pp: 295-307, 1998.

[37] Shukla, Anupam, Ritu Tiwari, and Rahul Kala, Real life applications of soft computing. Boca Raton: CRC Press, 2010.

[38] Reddy, I. Sathish, Shirish Shevade, and M. Narasimha Murty. "A fast quasi-Newton method for semi-supervised SVM", Pattern Recognition, vol. 44, no. 10, pp: 2305-2313, 2010.

[39] Marquardt, Donald W., "An algorithm for least-squares estimation of nonlinear parameters", Journal of the Society for Industrial & Applied Mathematics, vol. 11, no. 2, pp:431-441, 1963.

[40] Coskun, Nihan, and Tulay Yildirim, "The effects of training algorithms in MLP network on image classification", In Neural Networks, 2003. Proceedings of the International Joint Conference on, vol. 2, pp. 1223-1226. IEEE, 2003.

[41] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "The WEKA data mining software: an update", ACM SIGKDD Explorations Newsletter 11, no. 1, pp:10-18, 2009.

[42] Hansen, Lars Kai, and Peter Salamon, "Neural network ensembles", Pattern Analysis and Machine Intelligence, IEEE Transactions, vol. 12, no. 10, pp: 993-1001, 1990.

[43] Nakamura E, Inflation forecasting using a neural network. Economics Letters , vol. 86, pp: 373- 378, 2005.