Electrical Load Forecasting using Fast Learning Recurrent Neural Networks

Gul Muhammad Khan Atif Rashid Khattak Faheem Zafari Sahibzada Ali Mahmud

***Abstract*— A new recurrent neural network model which has the ability to learn quickly is explored to devise a load forecasting and management model for the highly fluctuating load of London. Load forecasting plays an significant role in determining the future load requirements as well as the growth in the electricity demand, which is essential for the proper development of electricity infrastructure. The newly developed neuro-evolutionary technique called Recurrent Cartesian Genetic Programming evolved Artificial Neural Networks (RCGPANN) has been used to develop a peak load forecasting model that can predict load patterns for a complete year as well as for various seasons in advance. The performance of the model is evaluated using the load patterns of London for a period of four years. The experimental results demonstrate the superiority of the proposed model to the contemporary methods presented to date.**

***Index Terms*— Load Forecasting, Neural Networks, Cartesian Genetic Programming, Neuro-evolution, Recurrent Neural Networks, Time Series Prediction.**

I. Introduction

The cost of electricity which a subscriber must pay is determined by various factors that are generally classified in terms of generation and distribution costs. The generation cost depends on both the means adopted for generation and the generation efficiency. The efficiency refers to maintaining balance between the demand and the generation of electricity by making sure that the generation is neither in excess nor less than the demand. Load forecasting is used to attain the balance between the demand and electricity generation which makes it an essential tool for the modern electricity system.

Load forecasting can be divided into three major categories: short-term forecasts (ranging from hourly to weekly basis) medium forecasts (weekly to one year), and long-term forecasts (more than a year) [1].

Energy storage on a large scale, even in modern times, is considered impractical and inefficient. The production therefore is maintained at a level to fulfill the demand, while simultaneously considering the increase in future demands of the electricity. This makes load forecasting extremely important as it allows for efficient planning of electricity supply from the power stations with future load increase in consideration.

Gul Muhammad, Atif Rashid Khattak, Faheem Zafari, and Sahibzada Ali Mahmud are with Centre for Intelligent Systems and Network Research, Electrical Engineering Department, UET Peshawar Pakistan (email: {gk502, atif.r.khattak, fahim.zafari, sahibzada.mahmud}@nwfpuet.edu.pk).

Various methods have been adopted to tackle the problem such as statistical analysis methods of time series analysis [2], Box Jenkins [3], linear regression and exponential damping in order to produce a forecasting model which is capable of predicting load forecasts years in advance. The data collected can not only be used for maintaining efficiency but for planning, dynamic market pricing and maintenance as well. Adaptive load forecasting techniques have also been developed that automatically change according to the varying load conditions. Artificial Intelligence (AI) based methods such as fuzzy logic system, expert systems and artificial neural networks (ANNs) are the most widely used approaches to load forecasting. Neural network methods, in particular, have been used in different hybrid approaches to develop an efficient load forecasting model.

This paper utilizes Cartesian Genetic programming (a type of evolutionary programming) to develop a Recurrent Neural Network (RNN) model with the major objective being the prediction of peak loads for the next day in advance using the historical load data of the last 10 days.

Section 2 describes a thorough literature review, that include a review of Computational Intelligence techniques for load forecasting, NeuroEvolution, Cartesian Genetic Programming (CGP), CGP evolved Artificial Neural Networks (CGPANN), Recurrent Neural Network (RNN) and Recurrent CGPANN in detail. Section 3 discusses the use of RCGPANN for load forecasting. Section 3 also highlights the experimental setup, along with the acquired results and analysis. The conclusion along with the possible future work is discussed in section 4.

II. Literature Review

A. Computational Intelligence Techniques for Load Forecasting

Genetic algorithms have been used in load forecasting to optimize the parameters that determine the forecast. Abbas et al. used SVM models totaling a number of seven models for the prediction of daily peak load demand [4]. Xu Tao et al. used Support Vector Machines for predicting short term load. A vector regression combined with a local prediction framework in a hybrid approach for load forecasting is suggested in El Atar et al. [5].

Artificial Neural Networks have also been used for load forecasting as they provide the capability of learning and can deal with the non-linear relationship nature between load and effects of the historical data. Pang Qingle et al. [6] suggested a neural network based on rough set. An ANN, which used 2 renowned algorithms known as Multi-Layer Perception (MLP) and Radial Basis Function (RBF) for forecasting the peak value of load without using the weather data is utilized in [7]. They concluded that better results were obtained using the MLP in their forecasting system. Using analyzable structure ANN for peak load forecasting as well as forecasting using scatter search based weighted average weather conditions is suggested in [8] [9]. Koushki et al. achieved Short-Term Load Forecasting (STLF) using a Local Linear Neuro-Fuzzy model [10]. 2 different non parametric procedures are combined to train the ANN structure and decide on the input for STLF in [11]. Mao et al. combined a self-organizing fuzzy neural network (SOFNN) learning method with a bi-level optimization method for their prediction model [12]. Khan et al. used Feed forward structure of CGPANN for short term load forecasting [13]. Bukhari et al. used ANN for the development of a STLF model for a 132/33Kv Sub Station [14].

The accuracy of an algorithm is usually determined using the Mean Absolute Percentage Error (MAPE). MAPE is generally considered an international standard to determine the performance of a load forecasting algorithm. It is given by:

$$MAPE = \frac{1}{N}\sum i = 1toN \left(\frac{|L_F - L_A|}{L_A}\right) \times 100$$

Where
L_F = Forecasted Load
L_A = Actual Load and,
N = Number of Season Days.

B. Neuro-Evolution (NE)

It is the process of incorporating artificial evolution techniques with the artificial neural networks. Different ANN aspects such as topology, node function and connection weights are evolved in NE. The genotype represents the network in the form of the array of numbers and the phenotype is the ultimate network. The desired characteristics of the phenotype are achieved through sustained evolution. It is possible to simultaneously evolve various parameters, but usually only a few parameters of the network are entertained (evolved). Evolving only the weights restricts the network's solution space. It is believed that in a restricted environment, novel solutions are not achievable. Therefore a range of methods have been devised in NE that evolve weights, topology or even both. This was further explained by Xin, Yao [15] who elaborated through experimentation that evolving both the topology and the weight produce better results than evolving either independently. In Symbiotic, Adaptive Neural Evolution (SANE) a population of neurons is evolved to form a neural network for a sequence oriented decision task [24]. Both the Cooperation and specialization in the population are promoted in this type of evolution, which results in a prompt, efficient genetic search and does not allow the network to converge to an inefficient solution [16]. Enforced Sub-Population (ESP) is the extension of SANE. In ESP a hidden layer subpopulation of neurons is evolved [17]. Neuro-evolution of Augmenting Topologies (NEAT) addressed three major problems in neuro-evolution which were: keeping track of genes that allows for crossover flexibility, speciation along with evolution from a simple structure and the subsequent complexification with the passage of time [23]. Performance wise NEAT is considered a major innovation. Lin Chen [18] proposed a hybrid extension to NEAT called Learning-NEAT. L-NEAT divides tasks into sub tasks which are then learned by incorporating back propagation into NEAT algorithm

C. Cartesian Genetic Programming (CGP)

Cartesian Genetic Programming is an innovative form of genetic programming that was introduced by Miller and Thompson [21]. It was initially aimed at the evolution of digital circuits. In CGP, the genotype has nodes represented in the form of genes (integer numbers). Each node has inputs and a function. The inputs can be programmed inputs and the outputs of each node can also act as an input to some other node. An activation function is also used which can be any mathematical function i.e. OR, AND, Sigmoid, Tangent-Hyperbolic, Step, Linear etc. The system output can be the output of any node in the genotype, indeed a system input can be an output as well. The genotype is an array of integers with a finite length, which is also called the chromosome. Mutation of either the node functions or the connection genes yields offspring.
The phenotype (the ultimate system) is yielded by following the referenced ordered links which exist in the graph . It means that the genes which form a path between the input and output are selected. In this process many genes are discarded (also termed as non-coding genes).

D. Cartesian Genetic Programming evolved Artificial Neural Network (CGPANN)

CGPANN is a neuro evolutionary technique known for providing much faster learning in comparison to various other neuro-evolution techniques [21]. CGPANN is based on CGP in which the nodes are replaced with artificial neurons that have weighted connections and non-linear activation functions. All attributes of neural networks such as its weights, topology, and node functions are encoded into genotype. The mentioned parameters are then evolved to the point where the best combination is achieved.
The evolutionary strategy (1+ λ) is used for generating the offspring. The mutation rate is set at 10% which means only 10 % of the genes are mutated to produce offspring. The inputs can be initial (program inputs) or intermediate (output of the program). The weights are generated randomly (ranging from -1 to +1) which are multiplied with inputs and summed up for each node. The output can be a subsequent input, the output of the program or any node.

E. Recurrent Neural Networks (RNNs)

RNN have a very dynamic behavior which makes them special neural networks. They differ from the feed forward neural networks due to the presence of feedback path (s) from the output back into the system as an input. The feedback path can exist for a single neuron as well as for a whole layer. The presence of feedback improves the learning ability of the network hence providing better results. The feedback paths are divided into various branches along with the presence of unit delay systems. This produces a non-linear dynamic behavior because of non-linear nature of neurons [31], which assists the recurrent neural networks in its storage function. Recurrent networks are sensitive and adaptive to the inputs from the past. In terms of modeling and controlling the non-linear systems, RNNs perform far superior to the feed forward neural networks. RNNs uses the temporal information of applied inputs for efficient prediction and classification. After the training, the interrelationship between the current inputs and internal states is processed for producing the output. The process of learning is a supervised process, in which the target value acts as the second source of information. The relevant interrelations in the input sequence are also highlighted by the target values. Time series is provided as an input to RNNs, however the target is either a trivial or non-trivial time series.

Units in an RNN form a directed cycle that produces a state of networking that is internal in nature, allowing it to exhibit dynamism [19]. These networks unlike feed-forward can process arbitrary sequence inputs due to their ability to access internal memory. RNN are applied in a wide variety of applications that vary from control automation to manufacturing, detection, time series prediction etc [25].

F. Recurrent Cartesian Genetic Programming evolved Artificial Neural Network (RCGPANN)

The significance of recurrent neural networks cannot be denied due to its capability in addressing a broader spectrum of dynamic and non-linear systems. Recurrent Cartesian Genetic Programming evolved artificial neural network was proposed by Khan et al. [29]. The algorithm is based upon using Cartesian Genetic Programming to evolve recurrent neural networks. The main characteriastics of this network is that it utilizes feedback path(s) from the output which is fed back as the input(s). RCGPANN uses direct encoding method by encoding the weights and topology function into the genotype. The genotype is then evolved until the optimum function, weights and topology are obtained. In RCGPANN, the offspring are produced using $1+\lambda$ principal and the value of λ is chosen to be 9.

Since RCGPANN is based upon TWEANN, it tends to act both constructive and destructive. During the evolution of the topological features, many new features might be added while old features might be removed. Mutation is used for the evolution of weights, functions, connection types, inputs and outputs. The connections which might be disabled due to mutation are not entirely removed, and are indeed saved so that they might be used in future generations.

One of the characteriastics of RCGPANN is that the neurons in RCGPANN are not totally connected and the inputs are not supplied to the neurons present in the input layer. Hence the topologies produced by RCGPANN are cost effective and efficiently timed.

Also, all the input layer neurons are not supplied with program inputs. This feature allows RCGPANN to produce topologies which is efficient in terms of timing and hardware implementation [30]. The genotype of a RCGPANN is made up of nodes which represent the ANN neurons. The inputs into each nodes belong to 3 different classes: inputs fed back from the output, the program inputs and the inputs from the previous nodes. The inputs may be recurrent or system inputs in the first layer of the RCGPANN genotype. However, the presence of feedback path(s) in other layers is dependent upon the fact whether selecting the feedback input, which is the node input, is random or not. Nodes are also either connected or disconnected provided that the connection value is either one or zero respectively. The weights assigned to the connections are randomly generated between -1 and +1, however the value of the weight of feedback input is always taken as +1. The activation functions which may be a sigmoid, linear, tangent hyperbolic functions; acts on the sum of the multiple of inputs and the weights of the connected inputs. The output at each nodes depends upon the activation functions.

The output obtained from the node is then either used as the output of the system or used an input to some other node. The genotype's output is taken from either a node output or program input. The genotype's output can also be used as a recurrent input.

The obtained RCGPANN genotype is continuously evolved until a desired level of fitness is achieved. The connections and state units weights are not mutated. The obtained genotype is then transformed into the RNN [29].

Fig 1(a) shows a RCGPANN node with weights, connection inputs, output and functions. Fig 1(b) represents inside views for the node shown in (a). I_1 and I_2 are the system inputs while R is the recurrent input which is fed back as input from the output. The inputs are multiplied with corresponding weights and are summed up. The activation function acts on the sum and produces the corresponding output. Fig 1(c) shows a general RCGPANN phenotype structure of a 3 input network the inputs being I_0 , I_1 and I_2 and three activation functions F_0 , F_1 and F_2. The genotype is represented by

$$\mathbf{[R, W_{R1}, I_2, W_{21}, F_{2,}\ I_{0,} W_{02}, I2, W_{22,} F_{1,} I_{1,} W_{13}, R, W_{R3}, F_2\ I_0, I_5, I_4\ ]}$$

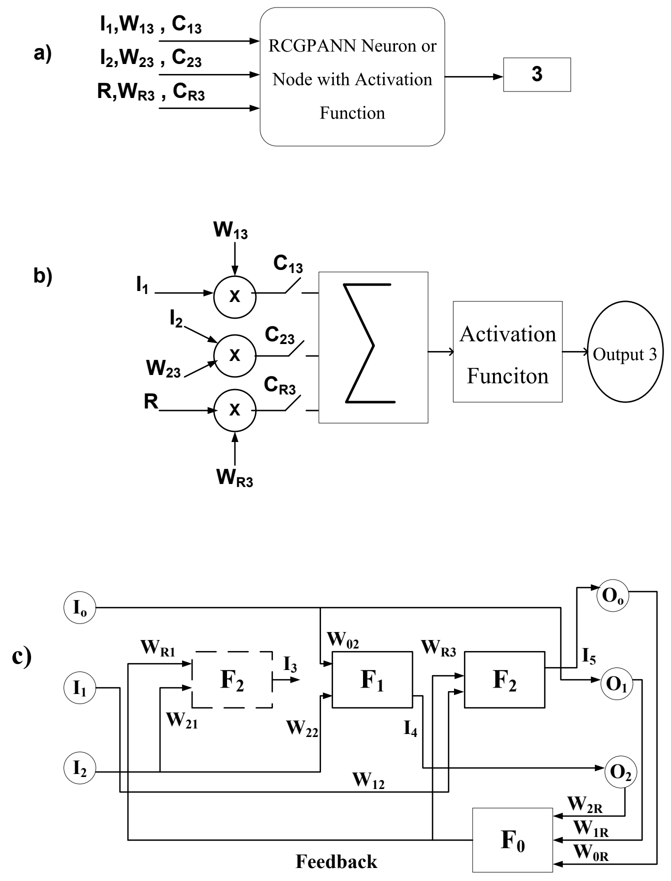

Figure 1: (a) RCGPANN node having two inputs (b) Internal View of RCGPANN for node in (a). (c) General RCGPANN phenotype structure.

III. RCGPANN APPLICATION FOR LOAD FORECASTING

A. Experimental Setup

For the experiment, the diurnal peak load (historical consumption data) is used as single input variable. The data of a complete year has been used for training the model. Initially, a population of 10 RCGPANN networks is produced. The number of inputs per node is selected to be 5. The Mutation rate to be used is 10% as it results in fast learning. Log-sigmoid function is chosen to be the activation function for producing the output. The number of RCGPANN rows is one; hence the number of columns and number of nodes are equal. After applying the inputs, its MAPE is calculated for every network generated and then compared with each other to select the network with optimum MAPE value. The selected network is used as the parent genotype for the next generation. The fittest genotype is mutated to generate further nine networks. This process continues until:

- Generation limit is attained
- The MAPE value becomes zero.

Ten independent evolutionary runs are performed for ten different network sizes, whereas each run is for 1-million generations which took 24 hours per run on each core-i7 cluster.

B. Results and Analysis

The RCGPANN network's performance was evaluated using the historical load data of United Kingdom National Grid which is in charge of maintaining the high-voltage electricity transmission system across Great Britain. Peak load information was extracted from the data which was originally collected as hourly load data. The annual as well as the seasonal data of 4 years was utilized. The data is highly fluctuating which makes it tedious to accurately predict the peak load. The data from the year 2006 was used for training purposes and the testing was done on data pertaining to the years 2007, 2008 and 2009. MAPE was used as the performance criterion. Subsequently, the genotype that exhibited the best fitness level was translated to a phenotype and then testing was performed for 2007, 2008 and 2009 on seasonal as well as annual basis.

The training along with testing was done repeatedly while the network size was varied from 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500 nodes. The final phenotype necessarily doesn't use all nodes of the network. Generally the phenotype is produced using only 5-10% nodes. Table I shows the training results in terms of different number of nodes. This shows the overall efficiency of the training process for the year 2006. The RCGPANN network with 100 and 250 nodes provide the optimum MAPE values. Table II shows the testing results of predictions for the year 2007, 2008, 2009 and independently for seasons of each year. In the annual results, the best result is obtained for the year 2008 for the networks of 100 and 250 nodes having a MAPE of only 2.19 percent; whereas for seasonal results, the best results are obtained for the autumn season. A MAPE value as low as 1.56 percent has been achieved. Table III shows a comparison of contemporary networks in terms of MAPE. The results produced by the RCGPANN are better in terms of prediction accuracy thus proving it's capability of accurate forecasting. This is because of the presence of the presence of feedback path and CGP algorithm. As stated earlier, the presence of the feedback enhances the ability of the network to learn quickly, so the proposed model outperforms all its predecessors.

Training Results: MAPE for various Node Numbers (Year-2006)										
Model	50	100	150	200	250	300	350	400	450	500
MAPE	2.43	**2.41**	2.43	2.42	**2.41**	2.44	2.43	2.47	2.45	2.44

Table I. Training
Results for Various Node Numbers (Neurons) scenario trained on the daily peak load for 2006, predicting 11th day load from the ten day's load data history.

Testing Results: MAPE for Various Node Numbers

Model	Year	50	100	150	200	250	300	350	400	450	500
Annual	2007	2.54	2.53	2.54	2.53	2.52	2.56	2.56	2.59	2.56	2.56
	2008	2.21	**2.19**	2.21	2.20	**2.19**	2.22	2.21	2.25	2.23	2.21
	2009	2.36	2.35	2.37	2.36	2.35	2.38	2.39	2.38	2.36	2.37
Autumn	2007	2.04	2.04	2.02	2.01	1.99	2.02	2.01	2.09	2.03	2.03
	2008	2.02	2.02	1.99	1.98	1.97	2.04	2.03	2.08	2.03	2.06
	2009	1.59	1.62	1.57	**1.56**	1.58	1.65	1.69	1.73	1.64	1.67
Winter	2007	2.75	2.71	2.72	2.72	2.69	2.78	2.73	2.79	2.77	2.78
	2008	4.05	4.02	4.03	4.02	4.01	4.09	4.06	4.07	4.05	4.11
	2009	3.36	3.35	3.35	3.34	3.34	3.39	3.41	3.36	3.34	3.41
Summer	2007	1.60	1.57	1.60	1.60	1.58	1.60	1.58	1.64	1.63	1.57
	2008	1.83	1.80	1.83	1.83	1.81	1.85	1.83	1.89	1.88	1.82
	2009	1.96	1.90	1.97	1.96	1.91	1.94	1.90	1.95	1.96	1.90
Spring	2007	3.02	3.02	3.05	3.05	3.04	3.10	3.17	3.07	3.05	3.10
	2008	2.24	2.24	2.25	2.25	2.25	2.25	2.28	2.24	2.24	2.25
	2009	2.63	2.65	2.66	2.67	2.67	2.62	2.66	2.59	2.59	2.62

Table II. Testing Results for Various Node Numbers (Neurons) scenarios for Various seasons of three years and yearly prediction.

Method	MAPE	Reference
Local Linear Model Tree	1.98%	[10]
Support Vector Machine	1.93%	[4]
Autonomous ANN	1.75%	[11]
Floating Search +SVM	1.70%	[20]
CGPANN	1.71%	[13]
ANN-Back Propagation	2.41%	[26]
GA based Adaptive ANN	1.94%	[27]
RCGPANN	**1.56%**	Implemented Algorithm

Table III. Comparison of RCGPANN with other methods

IV. Conclusion and Future Work

A Neuro-evolutionary technique called RCGPANN was used in this paper for daily peak load forecasting. The load data acquired was used to train the Artificial Neural Network for forecasting based on quarterly and annual basis and subsequently testing was done for peak load forecasting. The testing result shows that in comparison to previous models, RCGPANN produces more accurate results. Since the only input to the learning model is that of the peak load, in the future, other factors such as environmental conditions can be incorporated as inputs to produce even more Accurate results.

Acknowledgments: The authors would like to express their gratitude for the generous funding and resources provided by the **National ICT R & D Fund Pakistan.**

References

[1] S.A. Soliman, A.M Al-Kandari, "Electrical Load Forecasting: Modeling and Model Construction" *British library Cataloging*, 2010.

[2] J. Deng, and P.Jirutitijaroen, "Short-Term Load Forecasting Using Time Series Analysis: A Case Study for Singapore." *In proceedings of the 4th IEEE CIS & RAM 2010 - Session TA.*

[3] T.GÖNEN, "Load Forecasting using Box-Jenkins Methodology ", COMPEL: *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, Vol. 3 Issue: 1, pp.35 – 46, 1984.

[4] S. Abbas and A. M, "Electric load forecasting using support vector machines optimized by genetic algorithm." *In Proc. CIMCA '05*, pages 395–399, 2006

[5] E. E. El-Attar, J. Goulemas, Q. Wu, "Forecasting electric daily peak load based on local prediction." *In Power and Energy Society General Meeting*, pages 1–6, 2009.

[6] P. Qingle, Z. Min. "Very short-term load forecasting based on neural network and rough set." *In Proc. ICICTA '10*, pages 1132–1135, 2010.

[7] M. Ghomi, M. Goodarzi, M. Goodarzi, “Peak load forecasting of electric utilities for west province of Iran by using neural network without weather information.” *In Proc. UKSIM ’10*, pages 28–32, 2010.

[8] T. Matsui, T. Iizaka, Y. Fukuyama,. “Peak load forecasting using analyzable structured neural network.” *IEEE Power Engineering Society Winter Meeting*, 2:405, 2002.

[9] M. Kobayashi, T. Yukawa, Y. Kuze, T. Matsui, T. Iizaka, Y. Fukuyama, "Electric Load Forecasting using Scatter Search Based Weighted Average Weather Conditions," *International Joint Conference on Neural Network* '06. 2006. IJCNN.

[10] A. Koushki, M. maralloo, B. Hashemitabar, C. Lucas, “Load forecasting with the aid of neuro-fuzzy modeling.” *In Proceeding of Seventh International Conference on CSIT*, 2009.

[11] V.H. Ferreira, A.P da Silva, "Toward Estimating Autonomous Neural Network-Based Electric Load Forecasters," *IEEE Transactions on Power Systems*, vol.22, no.4, pp.1554-1562, Nov. 2007

[12] M.Huina, X.J.Zeng, G. Leng, Y.J. Zhai, J.A. Keane, "Short-Term and Midterm Load Forecasting Using a Bi-level Optimization Model," *IEEE Transactions on Power Systems*, vol.24, no.2, pp.1080-1090, May 2009.

[13] G.M. Khan, S. Khan, F. Ullah , "Short-term daily peak load forecasting using fast learning neural network," *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2011 , vol., no., pp.843-848, 22-24 Nov. 2011

[14] M. Buhari, S.S.Adam, “Short Term load forecasting using artificial Neural Networks ,” *Proceeding of International MultiConference of Engineering and Computer Scientists 2012 Vol I, IMECS 2012* March 14-16 2012 , Hong Kong.

[15] X. Yao “ Evolving artificial neural networks.” *In Proceedings of the IEEE*, volume 87(9), pages 1423–1447, 1999.

[16] D. E. Moriarty, “Symbiotic Evolution Of Neural Networks In Sequential Decision Tasks.” PhD Thesis Department of Computer Sciences, The University of Texas at Austin, 1997. 117. Technical.

[17] F. J. Gomez, R. Miikkulaine “Solving Non-Markovian Control Tasks with Neuroevolution (1999)” *In Proceedings of the 16th International Joint Conference on Artificial Intelligence*

[18] C. Lin, D. Alahakoon, "NeuroEvolution of Augmenting Topologies with Learning for Data Classification," *International Conference on Information and Automation*, 2006. ICIA 2006., vol., no., pp.367-371, 15-17 Dec. 2006.

[19] M. Graves, S. Liwicki, R. Fernandez, H. Bertolami, J. Bunke, J. Schmidhuber, “A Novel Connectionist System for Improved Unconstrained Handwriting Recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, 2009.

[20] X. Tao, H. Renmu, W. Peng, D. Xu. “Input dimension reduction for load forecasting based on support vector machine.” *In IEEE DRPT Conference*, pages 510–513, 2004.

[21] M. M. Khan, G. M. Khan, and J. F. Miller, “Evolution of neural networks using cartesian genetic programming.” *In IEEE CEC’10*, pages 1–8, 2010.

[22] J. F. Miller, P. Thomson. “Cartesian genetic programming.” *In Proc. of the 3rd European Conf. on Genetic Programming*, volume 1802, pages 121–132, 2000.

[23] K. O. Stanley, R. Miikkulainen (2002). "Evolving Neural Networks Through Augmenting Topologies". *Evolutionary Computation* **10** (2): 99–127.

[24] D. E. Moriarty, Risto Miikkulainen “Efficient Reinforcement Learning Through Symbiotic Evolution” *Recent Advances in Reinforcement Learning,* I94-224.

[25] P. A. Mastorocostas, I. B. Theocharia, “A stable learning algorithm for block-diagonal recurrent neural networks: Application to the analysis of lung sounds.” *IEEE Trans. on Systems, Man, and Cybernetics*, Part B, 36(2):242–254, 2006.

[26] L. Khan, K. Javed, S. Mumtaz, "ANN Based Short Term Load Forecasting Paradigms for WAPDA Pakistan", *Australian Journal of Basic and Applied Sciences*, 2010.

[27] J.H. Park, Y.M. Park, K.Y. Lee, “Composite Modeling for Adaptive Short-Term Load Forecasting”, *IEEE Transactions on Power Systems*, Vol. 6, Issue 2, pp. 450-57, May 1991.

[28] P. Smyth, D. Heckerman, M. Jordan, “Probabilistic Independence Networks for Hidden Markov Probability Models.” *Neural Computation*, 9:227–269, 1996.

[29] M. M. Khan, G. M. Khan, and J. F. Miller, “Efficient representation of recurrent neural networks for markovian/non-markovian non-linear control problems” *In proceeding of: FIT '10, 8th International Conference on Frontiers of Information Technology,* Islamabad, Pakistan, December 21-23, 2010.

[30] A. N. Refenes, M. Azema-Barac, L. Chen, and S. A. Karoussos, “Currency Exchange Rate Prediction and Neural Network Design Strategies,” *Neural Computing & Applications*, vol. 1, no. 1, pp. 46-58, Mar. 1993.

[31] S.F. Toha, M.O. Tokhi, “MLP and Elman Recurrent Neural Network modeling for the TRMS” *7th International Conference on Cybernetic Intelligent Systems,* pp. 1-6, 2008