World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:1, No:10, 2007

Auto-regressive Recurrent Neural Network Approach for Electricity Load Forecasting

Tarik Rashid, B. Q. Huang, M-T. Kechadi and B. Gleeson

***Abstract*—**this paper presents an auto-regressive network called the Auto-Regressive Multi-Context Recurrent Neural Network (ARMCRN), which forecasts the daily peak load for two large power plant systems. The auto-regressive network is a combination of both recurrent and non-recurrent networks. Weather component variables are the key elements in forecasting because any change in these variables affects the demand of energy load. So the AR-MCRN is used to learn the relationship between past, previous, and future exogenous and endogenous variables. Experimental results show that using the change in weather components and the change that occurred in past load as inputs to the AR-MCRN, rather than the basic weather parameters and past load itself as inputs to the same network, produce higher accuracy of predicted load. Experimental results also show that using exogenous and endogenous variables as inputs is better than using only the exogenous variables as inputs to the network.

***Keywords*—**Daily Peak Load Forecasting, Neural Networks, Recurrent Neural Networks, Auto Regressive Multi-Context Neural Network

I. INTRODUCTION

PREDICTION of energy load demand is vital in today's financial system. It is crucial because a correct estimation of energy can result in substantial savings for a power system. Once modeled appropriately it allows for the planning and designing of future plants, provides security and reliability, and reduces the operational cost of a power system. Several techniques have been implemented to solve the load-forecasting problem. These techniques can be categorized into factor analysis and time series [21, 20, 22] The factor analysis method is based on the determination of various factors, which influence the load demand, and working out their association with the load. However, the factor analysis method is incompetent as the evaluation of the factors involved is not easy. The time series method is based on the prediction of future load based on historical load. In the time series approach weather component variables are not involved in determining future load. Due to the limitations of this method, inaccurate and unstable predictions (forecasts) can be produced. Artificial neural networks deviate from the statistical models by their ability to map, in a fuzzy way, inputs to outputs. In this paper exogenous and endogenous input variables that are affecting the load are mapped to the load using neural network techniques.

The authors are in School of Computer Science & Informatics of University College Dublin, Belfield, Dublin 4, Ireland. Email: tarik.rashid@ucd.ie

The use of these networks [1, 25, 8, 7, 14, 18, 27] allows for the avoidance of the previous techniques' limitations by employing non-linear modeling and adaptation.

Artificial Neural Networks (ANN), as in [9, 1], are information processing paradigms simulated by the way biological nervous systems, such as the brain, process information. ANN can also be a form of multiprocessor computer system with straightforward process elements, a high degree of interconnection, easy scalar messages and adaptive relations between elements: ANNs are similar to people, as they learn from experience. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. There are mainly two types of neural networks: conventional neural networks and recurrent neural networks. Conventional neural networks, as in [9, 1], consist of three interconnecting layers; one input layer, one or more hidden layers and one output layer. Conventional neural networks allow signals to travel in one way only; from the input to the hidden layer and then to the output layer. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Conventional neural networks tend to be straightforward networks that associate inputs with outputs. Recurrent neural networks can have signals traveling in both directions by introducing loops into the network. These networks are very powerful but slower than the conventional networks, due to the loops, and can get extremely complicated. The simple recurrent network (SRN) [10] is an example of this type of network. The SRN is widely used by researchers, however, the network faces difficulties due to the architecture of the network itself: The architecture of the SRN includes the network memory, which consists of one context layer (relatively small) [6, 5, 26], the mapping of hidden layer neurons to the output layer neurons and an increased computation cost due to the need for more hidden neurons [2, 13]. The auto-regressive multi-context recurrent neural network is introduced to improve the speed of the training session due to a reduction of the recurrent connections, and is an appropriate method for approximating daily peak load.

O This paper is organized as follows: in the next section the autoregressive multi-context recurrent neural network is introduced, in section III learning algorithms are explained, in section IV we propose the forecasting system, in section V we display our experimental results, in section VI we propose an online forecasting system as our future work and finally we outline the conclusion of this paper.

II. Auto-regressive recurrent neural network

In [2], we have proved that the modified multi-context recurrent neural network (MCRN) overcomes the limitations of the SRN. The hybrid network introduced here, is a combination of the conventional neural network and SRN with MCRN [2] and is called the auto-regressive multi-context recurrent neural network (AR-MCRN). Two different AR-MCRN structures were designed, as can be seen from Figure 1a, the network is structured with two hidden layers on the same level; we called it AR-MCRN-a . However, in Figure 1b, the network is structured with two hidden layers on different levels, we called it AR-MCRN-b. In each topology, one hidden layer acts as a conventional neural network to the output layer while the other hidden layer acts as both a feed–forward to the output layer and a feed back to context layers. Logistic sigmoid transfer functions were used for all neurons in the hidden and linear transfer function was used for neuron in the output layer. This type of structure will improve the speed of the training session and is an appropriate method for approximating daily peak load [13, 24].

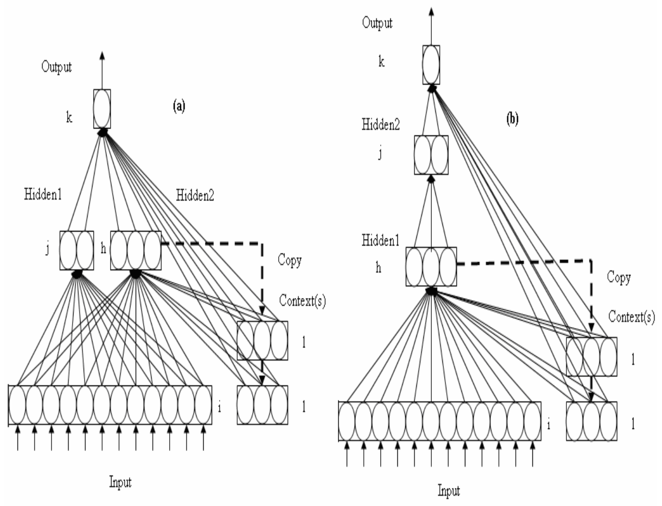


Fig. 1 a, displays the AR-MCRNN-a with hidden layers drawn in the same level, while b displays the AR-MCRNN-b with hidden drawn in different layer levels

III. Learning Algorithm

Neural networks are universally categorized in terms of their corresponding training algorithms: supervised, unsupervised and fixed weight. Supervised learning networks have been the mainstream of neural model development. The training data consists of numerous pairs of input/output training patterns, where the output pattern is the target output for the given input pattern. The learning will benefit from the support of a target. Examples of this are the conventional and simple recurrent networks [21, 9, 1]. For an unsupervised learning rule, the training set consists of input training patterns only. As a result the network is taught without the assistance of a target, such as the Kohonen network [16]. Fixed weight networks, as indicated by their name, cover fixed weights. No learning takes place; therefore, the weights cannot be modi┐ ed. An example of this type is the Hop┐ eld network [12]. For supervised learning networks, there are several learning techniques that are widely used by researchers. The main three are dynamic online learning, modi┐ ed back propagation and back propagation through time, all of which were used for our MCRANN [2, 23] depending on the application. Dynamic online learning is the most accurate amongst them, however, it is time consuming and slow due to the complexity of the computation. Modified back propagation is driven to include recurrent memory [2]. Modified back propagation was the quickest and produced accurate results.

IV. Forecasting System

The creation of a forecasting system can be described as follows: acquire and analyze the historical data, pre-process and normalise the data, choose the training and testing sets, choose the network architecture and its parameters, choose a suitable learning algorithm, and lastly implement the system.

A. *Historical Data*

Two historical data sets were collected to perform the forecasting task:

1. The first set that we term data set (A) was obtained from the EUNITE 2001 symposium, a forecasting competition. It reflects the behavior of the East Slovakia Electricity Corporation. This data recorded the load at half hour intervals every day from Jan 1997 to Jan 1999 and daily average temperature from Jan 1995 to Jan 1999.
2. The second set which we term data set (B), was obtained from the ESB Company. It reflects the behavior of the Electricity Supply Board in the Republic of Ireland. The data recorded the load, temperature, cloud rate, wind speed and humidity at fifteen-minute intervals every day from Jan 1989 to Jan 1999.

B. *Training and Testing Data*

The training and testing data sets for both data set (A) and (B) were cautiously selected to carry out the daily peak load forecasting and to estimate the performance of this new neural network. The training set consists of all data collected during the period January 1997 to December 1998 and the testing set concerns the data collected during January 1999.

C. *Input/Output Data Selection*

For this particular forecasting task the future load is a function of the accessibility of significant variables in both data sets. For data set (A) the future load is a function of the calendar, the status of the day (holidays), and the past and current change in the temperature T and past change in the load L. The future load in the data set (B) is a function of the calendar, the status of the day, the past and current change in the weather components (such as temperate T, cloud rate C, wind speed W and humidity H) and the past change in load L. In the following, details about the size and structure of the inputs of the MCRN network implementing each data set are given and explained. Note that as the two data sets contain different sets of parameters, the expression of the future load for each data set is given as follows:

1. The future (predicted) load for data set (A) is calculated from the difference between the historical values of load and its future values. The difference between the two is expressed as follows: $\Delta L_t = f($ past and current calendar; past and current social events; $\Delta T_t ,..., \Delta T_{t-n} ; \Delta L_{t-1} ,..., \Delta L_{t-n})$
2. The difference between the future and historical loads for data set (B) depends on a richer set of parameters than data set (A): $\Delta L_t = f$ (past and current calendar; past and current social events; $\Delta T_t ,..., \Delta T_{t-n} ; \Delta C_t ,..., \Delta C_{t-n} ; \Delta W_t ,..., \Delta W_{t-n} ; \Delta H_t$, ..., $\Delta H_{t-n} ; \Delta L_{t-1} ,..., \Delta L_{t-n}$).

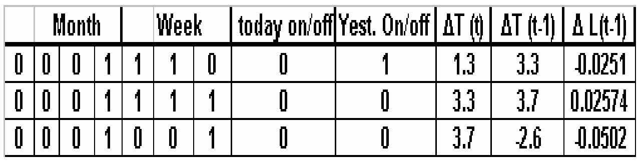
where t is the index of the day. The change in the load (difference between future and historical loads) and the change in weather components (temperature, cloud rate, wind speed, humidity) can be described as follows:

$$\Delta L_t = (L_t - L_{t-1}) / L_{t-1} ; \Delta T_t = T_t - T_{t-1} ;$$
$$\Delta C_t = C_t - C_{t-1} ;\ \Delta W_t = W_t - W_{t-1} ; \Delta H_t = H_t - H_{t-1} ;$$

According to the parameters recorded in each data set the size of the network input layer is 12 neurons for data set (A) and 18 neurons for data set (B). Let I_v denote an input neuron v. The following is the allocation of each input neuron of the network for data set (A):

1) Input neurons $I_1 ... I_4$ are allocated for the index of the month expressed in binary representation.
2) The next three input neurons I_5 , I_6 , I_7 represent the index of the week. Thus the network can identify the seasonal periods of the year and can also distinguish the days with high temperatures from those with low temperatures.
3) Input neuron I_8 indicates whether the forecasted day is a working day or a holiday.
4) Input neuron I_9 indicates whether the day prior to the forecasted day was a working day or a holiday. Usually this will affect the next day's load.
5) Input neuron I_{10} is for the change in temperature between the current day and the previous day: $\Delta T_t = T_t - T_{t-1}$.
6) Input neuron I_{11} is allocated for inputting the change in the temperature over the previous two consecutive days: $\Delta T_{t-1} = T_{t-1} - T_{t-2}$.
7) The last input neuron I_{12} is reserved for inputting the change in the load over the previous two days: $\Delta L_{t-1} = L_{t-1} - L_{t-2} / L_{t-2}$.

Figure 2, shows a sample of input data selected from data set (A).



	Month				Week		today on/off	Yest. On/off	ΔT (t)	ΔT (t-1)	Δ L(t-1)
0	0	0	1	1	1	0	0	1	1.3	3.3	-0.0251
0	0	0	1	1	1	1	0	0	3.3	3.7	0.02574
0	0	0	1	0	0	1	0	0	3.7	-2.6	-0.0502

Fig. 2 shows a sample of input data selected from data set (A) to the network

The network input layer for data set (B) consists of 18 neurons. In addition to the 12 inputs described above, another 6 input neurons are needed to represent other parameters recorded in this data set, such as wind speed, cloud rate, humidity, etc. Therefore, the first 11 neurons are exactly the same as for data set (A), and neuron 12 of network (A) is similar to neuron 18 of network (B). In the following we describe the additional neurons:

1) Input neuron I_{12} indicates the change in the cloud rate between the current day and the previous day: $\Delta C_t = C_t - C_{t-1}$.
2) Input neuron I_{13} indicates the change in the cloud rate over the previous two consecutive days: $I_{13} : \Delta C_{t-1} = C_{t-1} - C_{t-2}$.
3) Input neuron I_{14} indicates the change in wind speed between the current day and the previous *day:* $\Delta W_t = W_t - W_{t-1}$.
4) Input neuron I_{15} is allocated for inputting the change in wind speed over the previous two consecutive days: $\Delta W_{t-1} = W_{t-1} - W_{t-2}$.
5) Input neuron I_{16} indicates the change in humidity between the current day and the previous day: $\Delta H_t = H_t - H_{t-1}$.
6) Input neuron I_{17} indicates to the network the change in humidity over the previous two consecutive days: $\Delta H_{t-1} = H_{t-1} - H_{t-2}$.

For both networks with data sets (A and B), inputs $I_1 ... I_9$ are binary coded and inputs $I_{10} .... I_{18}$ are scaled between [0:1]. A binary representation is used for each group independently of the others. Alongside some other differences in parameter settings, which are described in the section above, both networks implementing data sets (A and B) have the same output layer, which consists of one neuron. The output of the networks is the current change of the daily peak load, which is the difference between the forecasted daily peak and the previous daily peak load: $\Delta L_t = L_t - L_{t-1} / L_{t-1}$. Both networks output is also normalised between 0 and 1.

The forecasting system which is described in [3] takes into account only the time and change in previous loads. In comparison, the technique offered in this paper considers more than just weather components. It considers the change in weather components for days, which are very close in time. This includes change in the load and details of the status of the day and calendar rather than just pure weather data. These changes are presented to the network as inputs and give the network a momentous enhancement in terms of accuracy and stability. The average error of the network performance dropped from approx. 4.5% to 1.9%. This is because the variation of the differences between the loads for two consecutive days is less than the differences between the loads factors themselves for two consecutive days. Consequently the network takes inputs in time series with values that are close to each other. This allows the network to learn more easily than if it was presented with inputs whose values are not close. The same remark applies to the other variables such as weather components. These types of differences between two parameter values that are close in time are shown in Figures 3 and 4. Figure 3 shows the variations in the daily average temperate for January 1997 and January 1998. Figure 4 shows the variations in the daily peak load for January 1997 and January 1998.

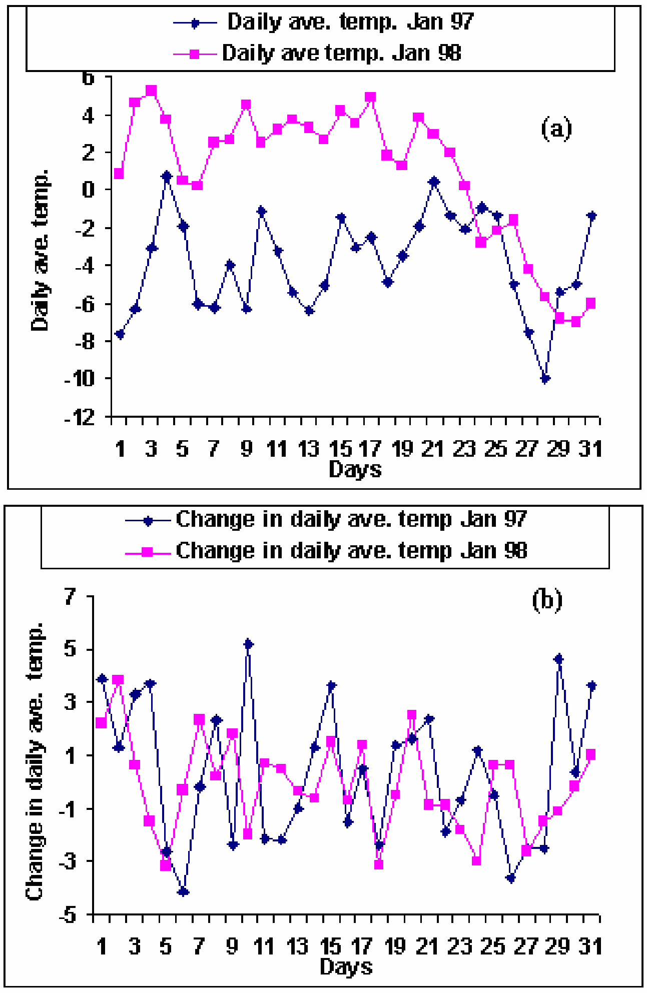
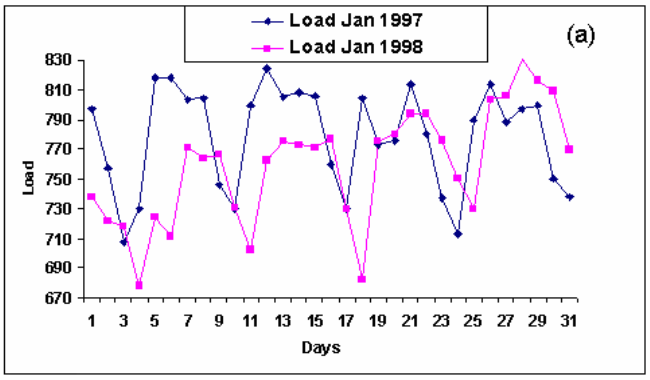


Fig. 3 (a) is the daily average temperate for the Jan 1997 and 1998, (b) is the difference between daily average temperature over consecutive days for Jan 1997 and 1998 (data set (A))



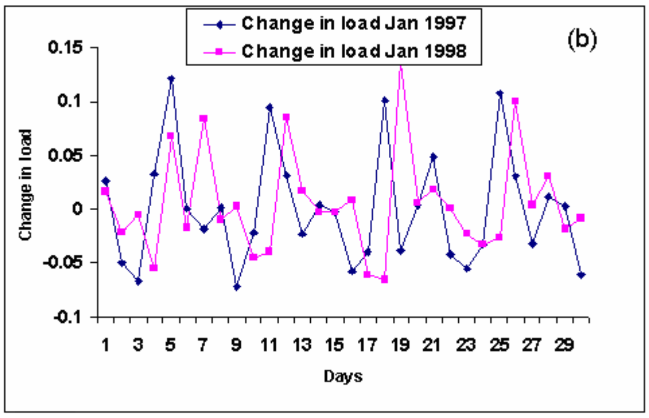


Fig. 4 (a) is the daily peak load for the Jan 1997 and 1998, (b) is the difference between daily peak load over consecutive days for Jan 1997 and 1998 (data set (A))

D. *Selection of Network Structure*

For each data set no exemption was made in terms of split models for weekend, weekday, and holiday or even for the days with odd behavior e.g. high temperature with load that did not decrease and low temperature with load that did not increase (no distinction was made for weekdays, weekends, winter season etc).

Three network structures were selected with different parameters. One network structure was selected for data set (A), because data set A has only one weather component variable (Daily average temperature). Whereas, two network structures were selected for data set (B), the first structure included only the daily average temperature and the second structure included all the weather components. The following details the network structures:

1) The AR-MCRNN-a structure and the AR-MCRNN-b structure for data set (A) each of which consisted of 12-2-3-2*3-1; 12 neurons, 2 neurons in the first hidden layer, 3 neurons in the second hidden layer, 2 context layers, each of which has 3 neurons, and 1 output neuron.
2) The AR-MCRNN-a structure and the AR-MCRNN-b structure for data set (B) each of which consisted of 12-2-3-2*3-1; 12 neurons, 2 neuron in the first hidden layer, 3 neurons in the second hidden layer, 2 context layers each of which has 3 neurons and 1 output neuron.
3) The AR-MCRNN-a structure and the AR-MCRNN-b structure for data set (B) each of which consisted of 18-3-4-2*4-1; 18 neurons in the input layer, 3 neurons in the first hidden layer, 4 neurons in the second hidden layer, 2 context layers each of which has 4 neurons and 1 output neuron.

These parameters relied profoundly on the size of the training and testing sets. Learning rates, momentum and the training cycles were varied. The type of activation function was a logistic function.

E. *Cross Validation, Training and Testing*

An effective algorithm is used for cross validation and to compute near optimal values for the network parameters such as learning rate, momentum, hidden neurons and the threshold value at which to stop training. Let TR denote the training set and TS the testing set used in this study (see Figure 5). The algorithm in general was as follows:

1. Invoke the training data set *TR* only.
2. Divide the training data set *TR* by n, so we have P_i validation set of data, for all $i = 1,2...n$ validation sets of data.
3. Let $P_i^{'}$ be the outcome of subtracting the P_i set from the *TR* set. Consider $P_i^{'}$ is a training set and P_i is validation set. For all $i = 1,2...n$.
4. Train the n networks independently, each with its training set $P_i^{'}$ and P_i test set. For all $i = 1,2...n$.
5. Compute the mean square error for each network MSE_i. For $i = 1,2...n$.
6. Optimize each network parameter (such as hidden neurons, learning rate, momentum etc). Repeat step 4.
7. Choose the best performance amongst the networks in terms of prediction and accuracy from step 5. Save the best MSE_i and the best weight connections as the optimized network mean square error $OMSE_i$ and weight connections OW_i.

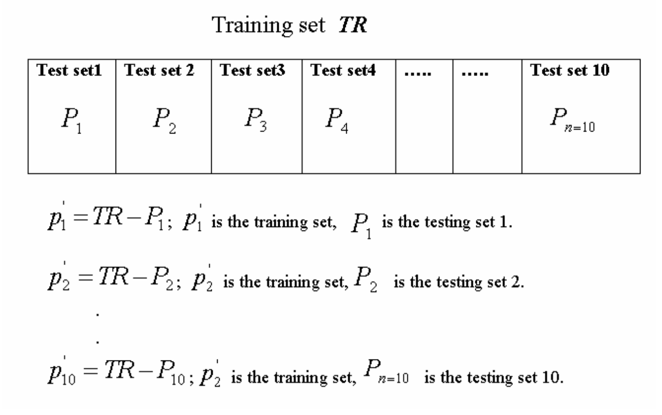
Testing of the network can be done in two ways:

1) Invoke the testing data set *TS*.
2) Load the network with the saved OW_i from above. Then, present the *TS* data set to the network. Obtain the forecasting results.

Or

1) Train the network with *TR*.
2) Stop the training when MSE of the network is equal to or less than the $OMSE_i$.
3) Present the *TS* data set to the network. Obtain the forecasting.

The two ways of testing are compared in terms of the forecasting results and the speed of convergence.


Training set *TR*

Test set1	Test set 2	Test set3	Test set4	.....	.....	Test set 10
P_1	P_2	P_3	P_4			$P_{n=10}$

$p_1^{'} = TR - P_1$; $p_1^{'}$ is the training set, P_1 is the testing set 1.

$p_2^{'} = TR - P_2$; $p_2^{'}$ is the training set, P_2 is the testing set 2.

.

.

$p_{10}^{'} = TR - P_{10}$; $p_2^{'}$ is the training set, $P_{n=10}$ is the testing set 10.

Figure 5 displays the cross validation procedures.

F. Complexity Computations of the AR-MCRNN

The multi-context layer in recurrent networks provides the potential for the network to store information about previous inputs. If one context layer is doing well for the task, it is possible that two or more context layers will construct the network better for a sequential task because they have more accurate information about the previous inputs. The number of context layers and the number of hidden layers, and neurons in each hidden layer are user specified. The common practice is to select these parameters so that the best achievable structure with as few potential parameters as possible is acquired. This cannot be very helpful, and, in practice, we have to experiment with different structures and evaluate their outcomes, to get the most appropriate neural network structure for the task to be tackled. In various applications, one or two hidden layers are adequate. The recommendation is to commence with a linear model, in order to facilitate neural networks with no hidden layers, followed by changing over to networks with one hidden layer but with no more than five to ten neurons. As a last step you should try two hidden layers. The number of weights for AR-MCRN-a can be calculated by the formula below:

$$w\ (o,h_1,h_2,I,c) = ch_2^2 + Ih_2 + 2h_2 + Ih_1 + h_1 \text{........(1)}$$

And the number of weights for the MCRNN-b can be calculated by

$$w(o,h_1,h_2,I,c) = ch_2^2 + Ih_2 + 2h_2 + h_1h_2 + h_1 \text{........(2)}$$

Where o, h_1, h_2, I, c are the number of output neurons, first hidden neurons, second hidden neurons, n input neurons and the number of context layers, respectively? As can be seen from the above two equations, they are very similar except that the fourth terms are different. Therefore, we expect that both AR-MCRN-a and ARMCRN-b can perform the task equally.

V. Experimental Results

The performance of the training and the validation of the network are evaluated by computing the sum of MSE_i averaged over the number of training and validation sets using the equation below:

$$MSE(perform) = \frac{1}{n}\sum_{i=1}^{n} MSE_{.i} \text{..............................(3)}$$

The error results of AR-MCRN-a obtained for load forecasting using the cross validation of 10 training and testing sets are shown in Table 1, using cross validation on data set A. The results from using the cross validation technique are very close to the actual forecasting errors produced by the network on the same data set as shown in Table 2. The second part of Table 2 shows the results for data set B, with only the change in the temperature component included as an input to the network. The last part of Table 2 displays the results for data set B, for which all the changes in weather components are included as inputs to the network. Obviously, the results shown in the last part presents better results in both accuracy in training and testing. Figures 6 and 7 display the load forecasting results of AR-MCRN-a and AR-MCRN-b for data set A and data set B, respectively, using only the change in daily average temperature component. While Figure 8 displays the forecasting results of AR-MCRN-a and AR-MCRN-b for data set B with the influence of all weather components. The evaluation of this network implementation of the load forecasting application is realised using two performance measures, namely the Mean Absolute Percentage Error (MAPE) and Maximum Error (MAX). The expressions of these two functions are given below, in equations (4) and (5):

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{Lr_i - Lp_i}{Lr_i}\right| \text{..(4)}$$

$$MAX = \max(Lr_i - Lp_i) \text{..(5)}$$

Where n, is the number of outputs forecasted from the network, Lr_i and Lp_i, are the target and the predicted values of the daily peak load, and i is the index of the day.

TABLE I
DISPLAYS THE RESULTS OF VARIOUS ERROR PERFORMANCES OF THE n NUMBERS OF TRAINING AND VALIDATION SETS ON THE AR-MCRNN-A FOR THE DATA SET (A)

Training & validation.sets	Data set (A)			
	MAX	MAPE	TrainingError	ErrorTest
(P'1,P1) sets	60.30	2.21	1.05E-04	0.019051276
(P'2,P2) sets	48.30	1.81	1.10E-04	0.092288542
(P'3,P3) sets	39.24	1.88	1.12E-04	0.063174716
(P'4,P4) sets	25.89	1.65	1.14E-04	0.057318069
(P'5,P5) sets	35.65	1.52	1.17E-04	0.071417407
(P'6,P6) sets	40.69	1.54	1.11E-04	0.034073225
(P'7,P7) sets	42.24	1.96	1.33E-04	0.06424313
(P'8,P8) sets	33.94	1.47	1.02E-04	0.1064
(P'9,P9) sets	55.93	1.70	1.15E-04	0.1499
(P'10,P10) sets	42.05	1.72	1.28E-04	0.092254687

TABLE II
DISPLAYS THE TRAINING AND TESTING DIFFERENT ERRORS OF OUR AR-MCRNN-A AND AR-MCRNN-B NETWORKS FOR BOTH DATA SETS (A AND B)

Data Set (A)	Only change in temperature component variable			
	MAX	MAPE%	TrainigError	TestingError
AR-MCRNN_a	26.85	1.58	1.14E-04	0.027512874
AR-MCRNN_b	33.72	1.57	1.33E-04	0.044681193
Data Set (B)	All the change of weather component (T, C, W, H) variables			
	MAX	MAPE%	TrainigError	TestingError
AR-MCRNN_a	214.37	1.99	9.06E-05	0.04511
AR-MCRNN_b	221.05	1.86	1.05E-04	0.1953
Data Set (B)	Only change in temperature component variable			
	MAX	MAPE%	TrainigError	TestingError
AR-MCRNN_a	304.59	2.38	1.88E-04	0.454378681
AR-MCRNN_b	300.64	2.02	1.56E-04	0.333530204

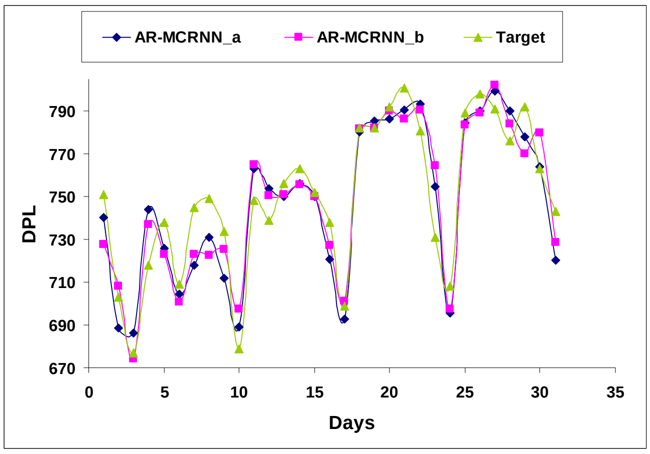


Fig. 6 displays the forecasting results of AR-MCRNN-a and AR-MCRNN-b for the data set (A) with only influence of the daily average temperature component variable

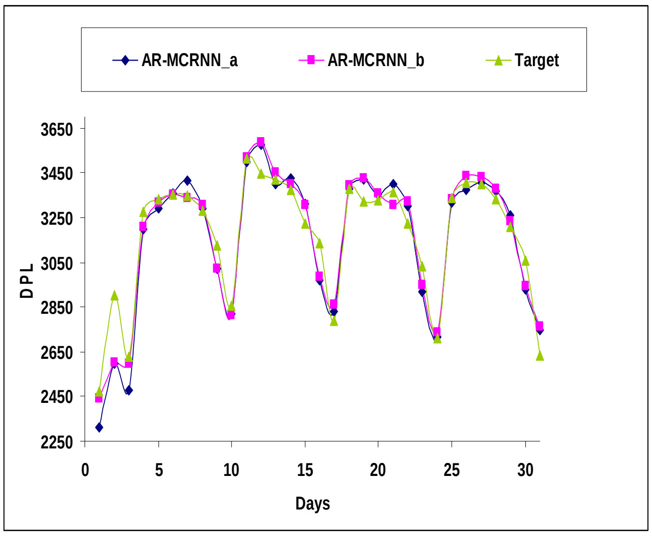


Fig. 7 displays the forecasting results of AR-MCRNN-a and AR-MCRNN-b for the data set (B) with influence of only the daily average temperature component variable

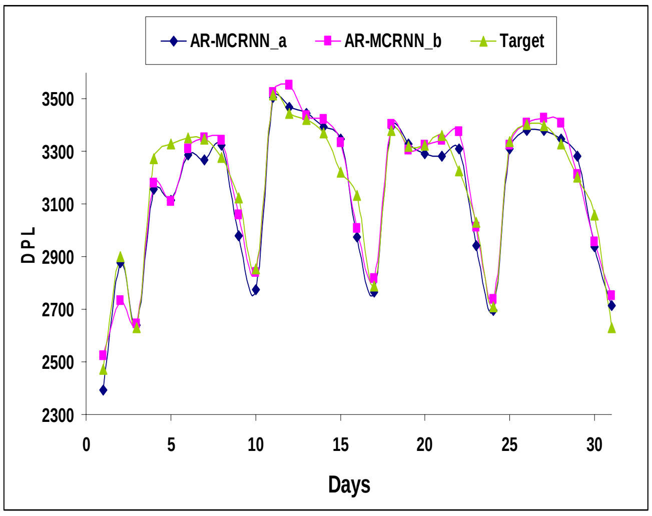


Fig. 8 displays the forecasting results of AR-MCRNN-a and AR-MCRNN-b for the data set (B) with influence of all weather components variables

VI. Conclusion

In this paper AR-MCRN networks are studied and used for daily peak electric load forecasting. Two historical data sets have been used on our networks. In this paper an effective approach for predicting the energy load is presented. The approach is mainly based on the neural network introduced initially in [10, 6, 5, 26, 2, 13]. Because the application of the initial network to the load-forecasting problem was not straightforward, some modifications and improvements in both the network structure and architecture were needed. AR-MCRN-a and AR-MCRN-b are designed to encode past histories and produce relatively equal accurate forecasting after short training periods. Furthermore, this paper also presented a different approach for modeling the load forecasting application. Weather components were identified and used in the model. The experimental results showed that the use of these components affected the network performance and therefore its output. More notably these components helped the network in the learning phase and made it easier and faster than without them.

In addition, the experimental results also showed a network presented with exogenous and endogenous inputs is better than a network presented with just exogenous inputs, as, in the first case, some relationships between various values of parameters were made clear. While, in the latter case, the days are implicitly the same if there is no information to the contrary. The results obtained in the first case were stable with higher precision than in the second case.

The main result of this paper is the development of well suited neural networks (AR-MCRN) to model the load forecasting application, and also the demonstration that the change in weather components over time leads to better performance than using current absolute weather components for the power plant peak load forecasting. Finally, the approach presented here compares favorably with other techniques proposed in [4, 11, 15, 17, 19], with maximum values of 1.5 in mean average percentage error.

VII. Future Work

Our future work will continue to study energy load forecasting using a new Auto-Regressive multi-context recurrent neural network. This network is characterised by the links from both hidden and output layers to the set of context layers. This network has previously been tested on other applications and has proved to be very competent when compared to networks in the same category such as Elman and Jordan networks.

We describe a methodology to take full advantage of this network's capabilities. This approach consists of two main phases: of-line training and online training. During of fine training, the network is trained with a few years' data. Then, from all this data a particular season is chosen and the network is re-trained using the weights of the first training run as initial weights. Again, at the end of this training session, the new weights are obtained and are used as initial weights to train for a particular month of that season.

The second phase has two main steps. The first step consists of selecting a day for which one wants to predict the load. According to the inputs of that day (i.e., temperature, weather parameters, etc.), a clustering technique is used to extract patterns (days) from the historical data that have “similar” features to that day. The network is then trained with these patterns. The second step starts just after the completion of the first step. It consists of inputting the selected day to the trained network and the output should correspond to the energy load of that day. Experimental results show that the network is very efficient and the prediction accuracy is very high.

References

[1] A. Bakirtzis, V.Petridis, and S. Klartizis, `A neural network short term load forecasting model for the greek power system', *In IEEE Tran. On Power Systems, 2 (11), 858--863, (1996).*

[2] B.Q.Huang, T.Rashid, and T.Kechadi, `A new modi┐ ed network based on the elman network', *in In Proceedings of IASTED International Conference on Arti┐ cial Intelligence and Application, ed., M. H. Hamza, volume 1 of ISBN: 088986-404-7, pp. 379--384, Innsbruck, Austria, (2004). ACTA Press.*

[3] W. Charytoniuk and M-S. Chen, `Very short-term load forecasting using neural networks', *In IEEE Tran. On Power Systems, 15(1), 1558--1572, (2000).*

[4] B.J Chen, M.W Change, and C.J. Lin, `Eunite network competition: Electricity load forecasting', *Technical report, In EUNITE 2001 symposium, a forecasting competition, (2001).*

[5] K.W.Yeung Dit-Yan, `A locally recurrent neural network model for grammatical interface', *in processing Proceedings of the International Conference on Neural Information Processing, pp. 1468--1473, (1995).*

[6] G. Dorffner, `Neural networks for time series processing', *In Neural Network World, 6, 447--468, (1996).*

[7] I. Drezga and S. Rahman, `Input variable selection for ann-based shortterm load forecasting', *In IEEE Trans. On Power Systems, 13(4), 1238--1244, (November 1998).*

[8] I. Drezga and S. Rahman, `Short term load forecasting with local an predictors', *In IEEE Trans. On Power Systems, 14(3), 844--850, (August 1999).*

[9] M. A. El-Sharkawi and Dagmar Niebur Editors, `Application of artificial neural networks to power systems', *In IEEE press, 96 TP 112-0, (1996).*

[10] J.L. Elman, `Finding structure in time', *In Cognitive Science, 14(2), 179--211, (1990).*

[11] D. Esp, `Adaptive logic networks for east slovakian electrical load forecasting', *Technical report, In EUNITE 2001 symposium, a forecasting competition, (2001).*

[12] J. J. Hop┐ eld, `Learning algorithms and probability distributions in feed-forward and feed-back networks', *In Proceedings of the National Academy of Sciences of the USA, volume 84, pp. 8429--8433, USA, (1987).*

[13] K. Kalaitzakis, G.S. Stavrakakis, and E.M. Anagnostakis, `Shot term load forecasting based on arti┐ cial neural networks parallel implementation', *In Eclectic Power System Research 63, pp. 185--196, Isanbul, Turky, (2002).*

[14] M.S. Kandil, S.M. El-Debeiky, and N.E. Hasanien, `Overview and comparison of long-term forecasting techniques for a fast developing utility', *In part I, Electric Power Systems Research 58, pp. 11--17, New York, (2001).*

[15] I. King and J. Tindle, `Storage of half hourly electric metering data and forecasting with arti┐ cial neural network technology', *Technical report, In EUNITE 2001 symposium, a forecasting competition, (2001).*

[16] T. Kohonen, Self-Organizing Maps, *Springer Verlag, Berlin, Heidelberg, 1995.*

[17] W. Kowalczyk, `Averaging and data enrichment: two approaches to electricity load forecasting', *Technical report, In EUNITE 2001 symposium, a forecasting competition, (2001).*

[18] A. Lendasse, M. Corttell, V.Wertz, and M. Veleysen, `Prediction of electric load using kohonen maps-application to the polish electricity consumption', *In the proceeding of the American Control conference Anchorage, pp. 8–10, AK, (May 2002).*

[19] L. Lewandowski, F. Sandner, and P. Portzel, `Prediction of electricity load by modeling the temperature dependencies', Technical report, In EUNITE 2001 symposium, a forecasting competition, (2001).

[20] A. D. Papalxopoulos and T. C. Hiterbe, `A regression-based approach to short-term load forecasting', *In IEEE Tran. On Power Systems, 5(4), 1535--1547, (1990).*

[21] D. Park, M. Al-Sharkawi, R. Marks, A. Atlas, and M. Damborg, `Electric load forecasting using an arti┐ cial neural network', *In IEEE Tran. On Power Systems, 6(2), 442--449, (1991).*

[22] S. Rahman, `Formulation and analysis of a rule-based short-term load forecasting algorithm', Proceedings of the IEEE, 39, pp.161--169, (1994).

[23] T. Rashid, B.Q. Huang, and M.-T Kechadi, `A new simple recurrent network with real-time recurrent learning process', *In the14th Irish Arti┐ cial Intelligence and Cognitive Science (AICS'03), volume 1, pp. 169--174, Dublin, Ireland, (2003).*

[24] T. Rashid and T. Kechadi, `A practical approach for electricity load forecasting', *In the proceeding WEC'05, The 3rdWorld Enformatika, volume 5, pp. 201--205, Isanbul, Turky, (2005). ACTA Press.*

[25] D. Srinivasan, `Evolving arti┐ cial neural networks for short term load forecasting', in In Neuro computing, volume 23, pp. 265--276, (1998).

[26] William H. Wilson, `Learning performance of networks like elman's simple recurrent netwroks but having multiple state vectors', *Workshop of the 7th Australian Conference on Neural Networks, Australian National University Canberra, (1996).*

[27] B-L. Zhang and Z-Y. Dong, `An adaptive neural-wavelet model forshort term load forecasting', *In Electric Power Systems Research 59, pp. 121--129, New York, (2001).*