01 Jun 2009

Comparison of Feedforward and Feedback Neural Network Architectures for Short Term Wind Speed Prediction

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Richard L. Welch

Stephen M. Ruffing

Recommended Citation

G. K. Venayagamoorthy et al., "Comparison of Feedforward and Feedback Neural Network Architectures for Short Term Wind Speed Prediction," *Proceedings of the International Joint Conference on Neural Networks, 2009. IJCNN 2009*, Institute of Electrical and Electronics Engineers (IEEE), Jun 2009.
The definitive version is available at https://doi.org/10.1109/IJCNN.2009.5179034

Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Comparison of Feedforward and Feedback Neural Network Architectures for Short Term Wind Speed Prediction

Richard L. Welch, Stephen M. Ruffing, and Ganesh K. Venayagamoorthy

Abstract—**This paper compares three types of neural networks trained using particle swarm optimization (PSO) for use in the short term prediction of wind speed. The three types of neural networks compared are the multi-layer perceptron (MLP) neural network, Elman recurrent neural network, and simultaneous recurrent neural network (SRN). Each network is trained and tested using meteorological data of one week measured at the National Renewable Energy Laboratory National Wind Technology Center near Boulder, CO. Results show that while the recurrent neural networks outperform the MLP in the best and average case with a lower overall mean squared error, the MLP performance is comparable. The better performance of the feedback architectures is also shown using the mean absolute relative error. While the SRN performance is superior, the increase in required training time for the SRN over the other networks may be a constraint, depending on the application.**

I. Introduction

THE limited existing reserves of fossil fuels and the harmful emissions associated with them have led to an increased focus on renewable energy sources in recent years. Among renewable energy sources, wind energy is the one with the lowest cost of electricity production [1], but is feasible only as long as weather conditions allow. To maintain economical power dispatch of wind generated electricity, it is important to be able to make short term predictions of future wind speed, which directly affects generation capacity. Without this ability, a wind farm operator is prone to allocate more generation units or supplemental energy reserves than necessary in order to ensure budgeted electricity outputs are met [1], with an end result of increased operating costs.

Numerous examples exist in literature of training neural networks (NNs) to make short term wind speed predictions. The neural network types utilized in these studies generally consisted of either the feedforward multi-layer perceptron (MLP) network [2], [4]-[6] or recurrent neural network (RNN) [7], [8] structure. Research has shown the recurrent neural network structure to be effective for time-series data forecasting [7] and, therefore, it is assumed to be the best architecture for short term wind speed predictions.

Manuscript received March 9, 2009. This work was supported in part by a U.S. Department of Education grant: GAANN # P200A070504, a NSF grant EFRI # 0836017, and Missouri S&T's Intelligent Systems Center.

R. L. Welch, S. M. Ruffing and G. K. Venayagamoorthy are with the Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, Missouri S&T, Rolla, MO 65409 USA (e-mail: rwelch@ieee.org, smrr34@mst.edu & gkumar@ieee.org).

In this study, the short term wind speed prediction abilities of an MLP, RNN, and simultaneous recurrent neural network (SRN) are investigated. The training algorithm used is particle swarm optimization (PSO). The resulting neural networks trained with this training algorithm are compared.

Previous work has shown temperature to be the most important meteorological parameter in predicting short term future wind speed [2]. Humidity and current wind speed have also been identified as key indicators. The data used to train the neural networks in this paper, therefore, incorporates the following three inputs (in addition to a bias value) recorded by instruments on or near the 82 meter tall M2 tower of the National Renewable Energy Lab (NREL)'s National Wind Technology Center (NWTC) site [3] located 5 miles south of Boulder, Colorado:

- Current wind speed measured at 80 m in m/s
- Current temperature at 2 m in degrees C
- Current percent relative humidity

The training data includes values for each of these inputs in one minute intervals, with each data point representing the mean of readings taken every two seconds during that minute. This training data is used to optimize the values of synaptic weights of each neural network in order to provide the best prediction of 80 m wind speed fifteen minutes into the future.

The remainder of this paper is organized as follows: A detailed examination of three neural networks used is given in section II; in section III, the training method is explained in detail, with the results given in section IV. Section V gives the conclusions of this work.

II. Neural Networks

Neural networks comprise one of the five main computational intelligence paradigms [9], and are known as universal approximators. The first (and most popular) network is the MLP network; the second is the RNN; finally, the third neural network that is investigated is SRN. In each case, the network inputs are as discussed in the previous section, and the output is just the wind speed 15 minutes into the future. All input values are normalized by their maximum value during the week's worth of training data prior to being presented to the network.

A. Multi-Layer Perceptron Network

The MLP network is a member of the feedforward network architecture, and is the simplest of the networks under investigation. In this network, there are 3 layers, each composed of neurons. The 3 layers are the input, hidden, and

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

output layers. The input layer (with a linear activation function) is fed the input values which are then multiplied by an input weight matrix *W*, passed through the hidden layer (using the sigmoid activation function), multiplied by an output weight matrix *V*, and finally fed to the output layer (which uses a linear activation function). A diagram of the MLP used is shown in Fig. 1.

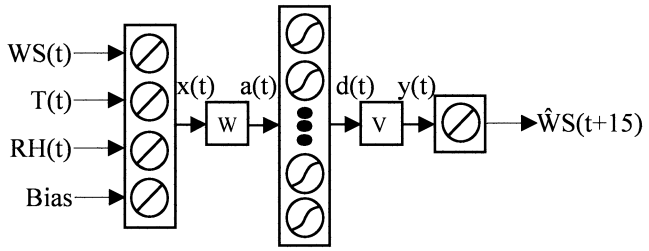

Fig. 1. Diagram of a MLP showing inputs and outputs. *WS*(*t*), *T*(*t*), *RH*(*t*), and $\hat{W}S(t+15)$ are the wind speed, temperature, and relative humidity at time *t*, as well as the predicted wind speed at time *t*+15, respectively.

The equations (1) through (4) are used to calculate the output *y* from a given input *x*.

$$x(t) = [WS(t), T(t), WH(t), Bias] \tag{1}$$

$$a(t) = W \times x(t) \tag{2}$$

$$d(t) = \frac{1}{1 + e^{-a(t)}} \tag{3}$$

$$y(t) = V \times d(t) \tag{4}$$

B. Recurrent Neural Network

The RNN is a member of the feedback architecture, and as mentioned this type of architecture has been shown to excel at time series prediction. The RNN is similar to the MLP in general structure except that it contains a feedback loop (with unit delay) from some later stage of the network back to the input layer. In this study, an Elman network is used which takes the output from the hidden layer; another type of RNN is the Jordan network which takes its feedback from the output layer. This feedback is stored in another layer called the context layer which allows the network to retain an internal memory. A diagram of the Elman RNN is shown in Fig. 2.

The Elman structure is chosen over the Jordan network in this study due to the hidden layer being wider than the output layer. This wider layer allows more values to be fed back to the input, thus allowing more information to be available to the network.

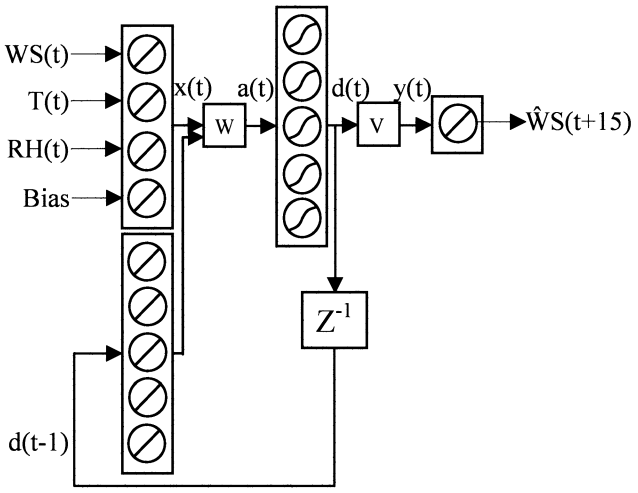

Fig. 2. Diagram of an Elman RNN. *WS*(*t*), *T*(*t*), *RH*(*t*), and $\hat{W}S(t+15)$ are the wind speed, temperature, and relative humidity at time *t*, as well as the predicted wind speed at time *t*+15, respectively.

To compute the output of an RNN, the same equations from an MLP are used, except for (1) which is replaced by (5).

$$x(t) = [WS(t), T(t), WH(t), Bias, d(t-1)] \tag{5}$$

C. Simultaneous Recurrent Neural Network

As with the RNN, the SRN is a member of the feedback architecture. However, the SRN differs from the RNN in that the feedback portion does not contain a delay unit and instead this feedback value is taken directly from the later stage of the network. Again as with the RNN, the SRN can come in either an Elman or Jordan configuration. In this research, the Elman approach is used. A diagram of the Elman SRN is shown in Fig. 3.

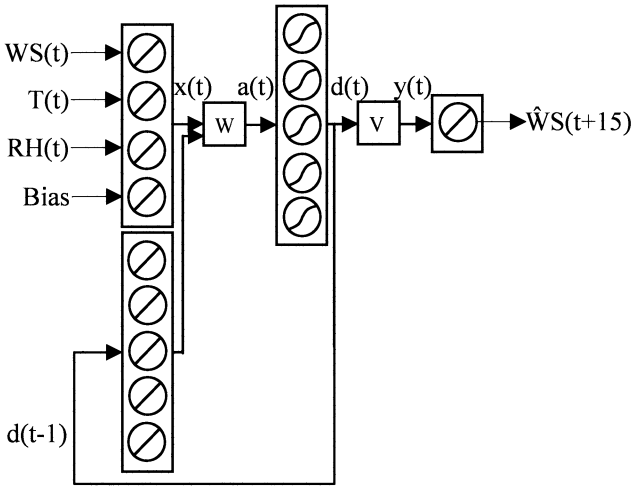

Fig. 3. Diagram of an Elman style SRN. *WS*(*t*), *T*(*t*), *RH*(*t*), and $\hat{W}S(t+15)$ are the wind speed, temperature, and relative humidity at time *t*, as well as the predicted wind speed at time *t*+15, respectively.

One major difference between an SRN and the RNN is that since the feedback does not contain a delay, the input values must be propagated several times through the network until the output reaches equilibrium. This process of repeatedly applying the inputs is called an *internal*

recurrence, and is required because the feedback value depends directly on the output of the hidden layer (without delay). Because of these internal recurrences, the same equations used to obtain an output for the RNN can be used for the SRN, except that they must be repeated multiple times until the output y in (4) stabilizes. To insure stability of the output, internal recurrences are executed until two successive outputs differ by no more than 0.01 or 20 iterations have been reached.

In this study, each network has 4 input neurons (one for each of the inputs and also a bias value) and 1 neuron in the output layer since the network will produce one output value. However, to keep the number of synaptic weights the same (and hence of amount of information that each network contains), the MLP has 10 neurons in its hidden layer while the RNN and SRN each have 5 neurons in the hidden layer, which also means that the context layers are 5 neurons wide. This gives W a size of 4×10 and V a size of 10×1 for the MLP case; the RNN and SRN both have a W of size 9×5 and a V of size 5×1; in all cases, the total number of weights is 50. The hidden layer size is chosen rather arbitrarily since the optimum number cannot be known without trial and error or analyzing the degree of freedom of the training data [10].

III. Training Procedures

In the case of each of the previously mentioned neural networks, both the input and the output weights need to be trained in order to allow the network to provide an adequate output. In this study, particle swarm optimization [11] is used for this process and is implemented in Matlab. Previous work has shown that the PSO algorithm demonstrates superior performance compared to backpropagation algorithms when used to train feedforward neural networks such as an MLP [12].

PSO is a computational optimization technique that utilizes a group of particles that fly through a multidimensional problem space, where each particle represents a potential solution. The number of dimensions in the problem space is equal to the number of parameters that need to be optimized. Because each of the networks being investigated has the same number of synaptic weights, the number of dimensions of each particle in each case is 50.

The implementation used in this study is the standard canonical form, with velocity update given by (6) and position update given by (7). In each equation, e is the particle number and d is the particle dimension. Additionally, the fitness value used during the PSO process is given in (8). In (8), p is the number of training points in the training set.

$$\begin{aligned} v_{ed}(t+1) &= w \times v_{ed}(t) \\ &+ c_1 \times rand_1 \times (p_{best,ed} - x_{ed}(t)) \\ &+ c_2 \times rand_2 \times (g_{best,d} - x_{ed}(t)) \end{aligned} \tag{6}$$

$$x_{ed}(t+1) = x_{ed}(t) + v_{ed}(t+1) \tag{7}$$

$$Fitness = \frac{\sum_{i=1}^{p} (y_{target} - y_{actual})}{p} \tag{8}$$

Before the actual training of the neural networks takes place, the parameters of the PSO algorithm (w, c_1 and c_2) as well as the number of particles used for the PSO optimization are optimized as is done in [13]. This optimization process results in using 30 particles with an initial value of w found to be 0.7492 (this value is linearly decreased to half its initial value at the final iteration), while the optimal values of c_1 and c_2 are found to be 0.8107 and 2.5000, respectively. After all PSO parameters are optimized, each of the previously mentioned networks are trained with PSO for 300 iterations with all of the optimal settings and using the 10080 points of training data from the week of 11/10/2008 through 11/16/2008, with each data point corresponding to the one minute average of readings taken every two seconds. This training is repeated 20 times in order to average out variations in performance that occur due to the stochastic aspects of the PSO algorithm.

IV. Results

The resulting worst, average, and best MSE of each network trained with PSO, along with the time required to run the entire optimization process for each network is given in Table I.

TABLE I
RESULTS OF TRAINING EACH NEURAL NETWORK USING PSO

Network Type	Worst MSE ($\times 10^{-4}$)	Average MSE ($\times 10^{-4}$)	Best MSE ($\times 10^{-4}$)	Run Time Required
MLP	6.504	6.252	6.088	19.01 hours
RNN	6.445	5.959	5.655	20.04 hours
SRN	6.263	6.143	5.922	55.21 hours

Examples of how the fitness of the best g_{best} particle improves over time for each network are given in Fig. 4.

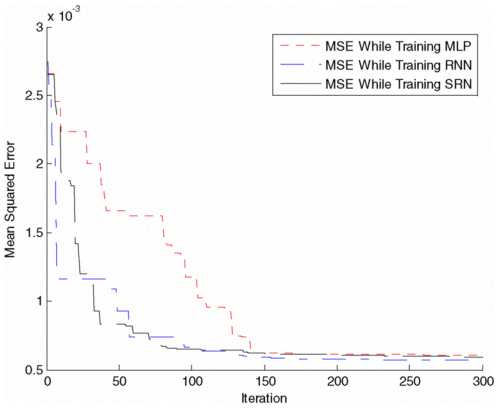

Fig. 4. Mean squared error of the best g_{best} particle for training all networks.

The performance of the best of each network along with the absolute relative error (ARE) as calculated in (9) is also compared. A sampling of each for a small selected time frame is shown in Figs. 5-10.

$$ARE = \left| \frac{WS(t) - \hat{WS}(t)}{WS(t)} \right| \quad (9)$$

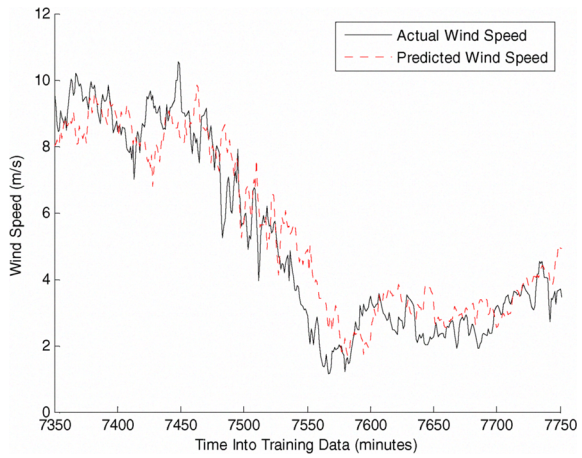

Fig. 5. The actual performance of the best MLP network trained using PSO, showing both actual wind speed and predicted wind speed.

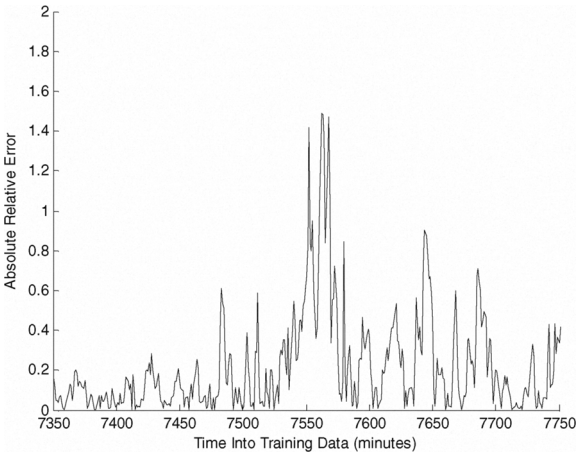

Fig. 6. The absolute relative error of the best MLP network trained using PSO.

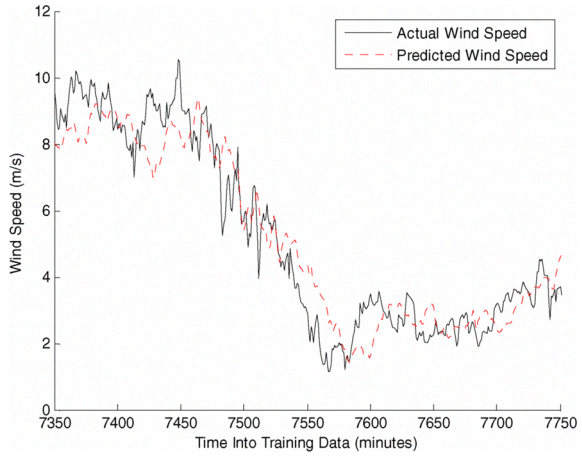

Fig. 7. The actual performance of the best RNN network trained using PSO, showing both actual wind speed and predicted wind speed.

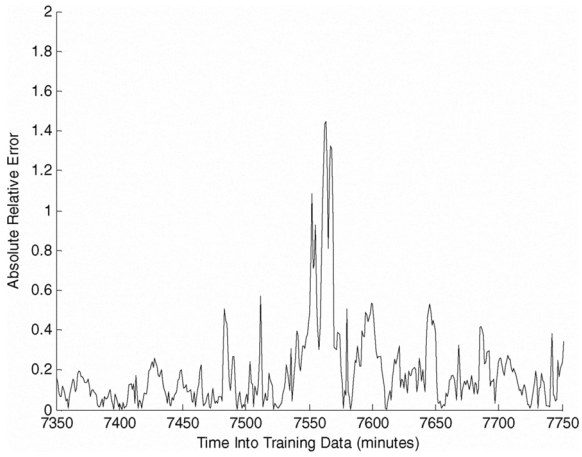

Fig. 8. The absolute relative error of the best RNN network trained using PSO.

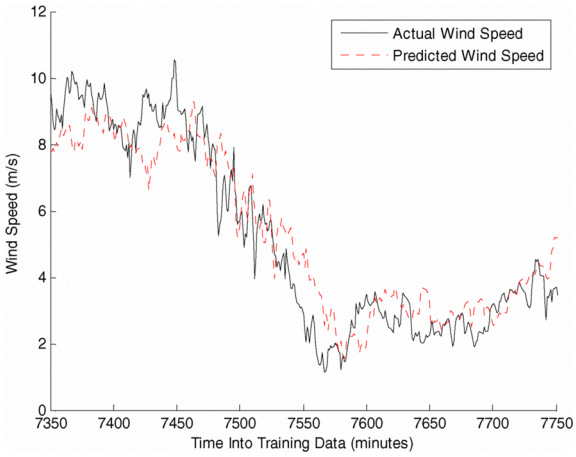

Fig. 9. The actual performance of the best SRN network trained using PSO, showing both actual wind speed and predicted wind speed.

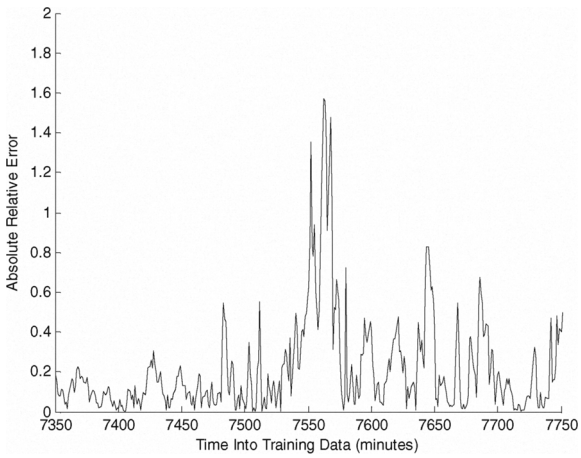

Fig. 10. The absolute relative error of the best SRN network trained using PSO.

In Figs. 5, 7 and 9, the performance of the MLP, RNN, and SRN networks is shown for a sample of the training data, respectively. From these figures, it is shown that while the performance is similar with each network, the feedback architectures (RNN and SRN networks) slightly outperform the MLP. In Figs. 6, 8, and 10, the ARE of each network

over this sample period is shown. In particular, it is noted that the feedback architectures again outperform the MLP, especially between minutes 7600 and 7700. During this time frame, the magnitude of the ARE is lower for the feedback architectures than it is for the MLP.

In addition to testing the performance of each network using the training data set, the networks are tested against data outside of the training set, in this case using data from 5/10/2008 through 5/16/2008. Examples of this are shown in Fig. 11-16, in which the results (and ARE) for each network are given for a subset of the testing period.

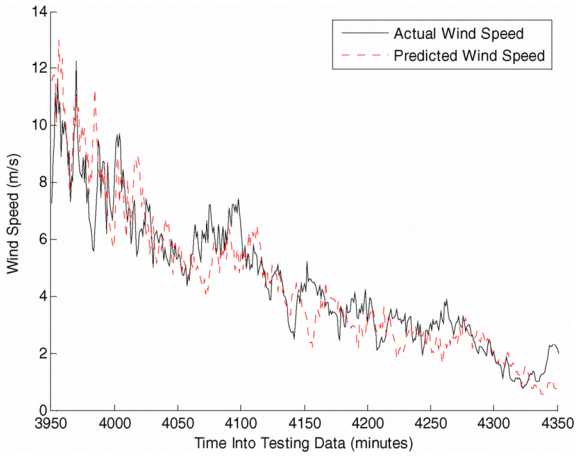

Fig. 11. The actual performance of the best MLP network trained using PSO, showing both actual wind speed and predicted wind speed for a period in a dataset for a week in May, 2008.

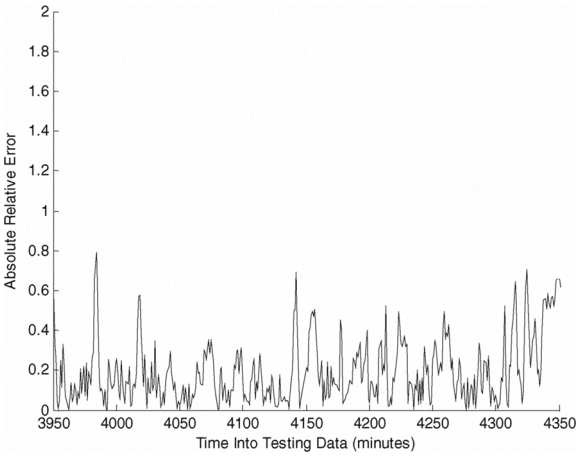

Fig. 12. The absolute relative error of the best MLP network trained using PSO.

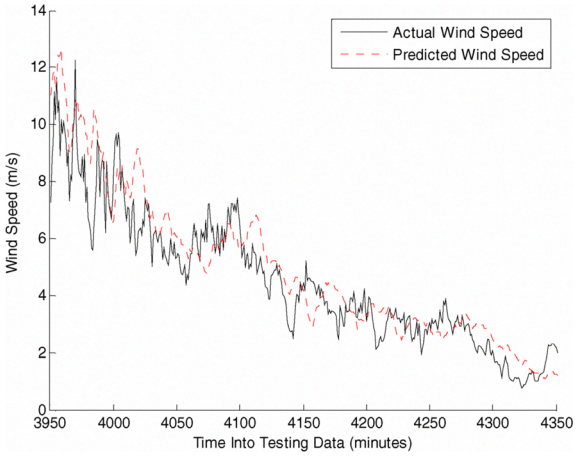

Fig. 13. The actual performance of the best RNN network trained using PSO, showing both actual wind speed and predicted wind speed for a period in a dataset for a week in May, 2008.

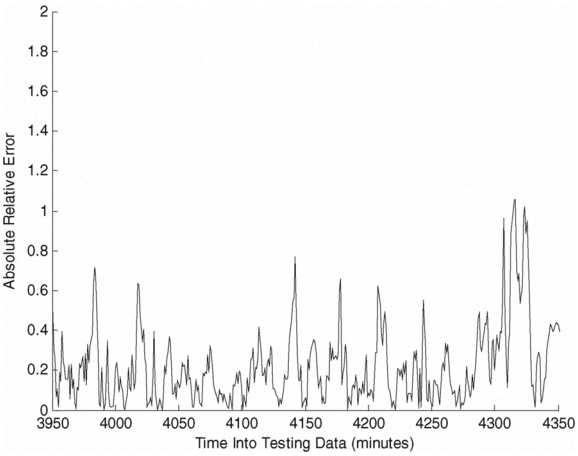

Fig. 14. The absolute relative error of the best RNN network trained using PSO.

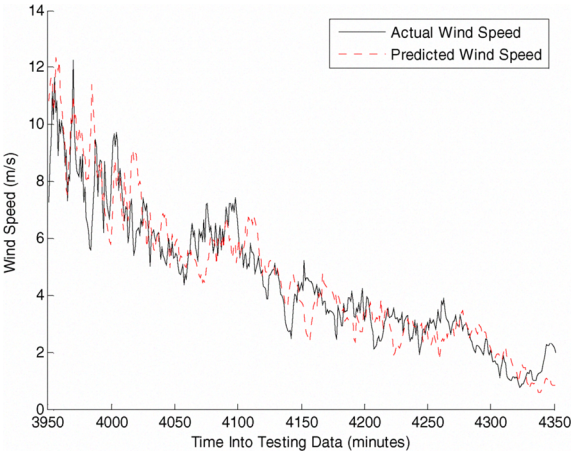

Fig. 15. The actual performance of the best SRN network trained using PSO, showing both actual wind speed and predicted wind speed for a period in a dataset for a week in May, 2008.

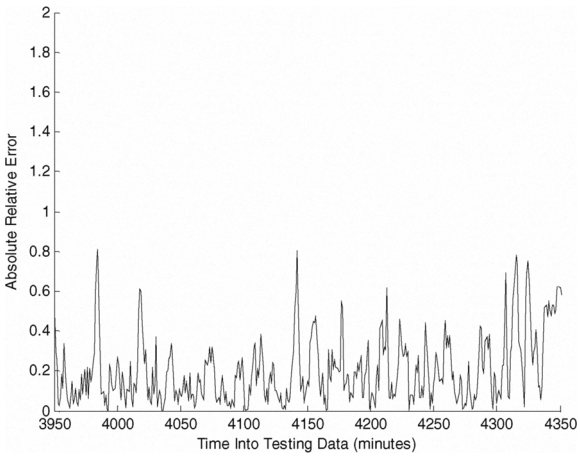

Fig. 16. The absolute relative error of the best SRN network trained using PSO.

As in the case with the training data, the data shown in Figs. 11, 13 and 15 show that the performance of the feedback networks outperforms the MLP. Figs. 12, 14 and 16 show the ARE of each network.

Finally, the mean absolute relative error is given in Table II for each of the networks for the entirety of both the training and testing data sets.

TABLE II
MEAN ABSOLUTE RELATIVE ERROR FOR EACH DATA SET

Network Type	Training Data	Testing Data
MLP	0.3847	0.5038
RNN	0.3892	0.4354
SRN	0.3795	0.4544

V. CONCLUSIONS

The ability to perform short term wind speed prediction accurately is a very useful feature for the electric power industry. This provides wind farm operators the ability to anticipate wind power output, and participate in the electricity market and operation of the power network accordingly. The results obtained in this study show that while all neural networks investigated have the potential for short term wind speed prediction, the best results are generally obtained from the recurrent neural architectures, especially on data outside the training range. However, with this increase in accuracy comes an increase in training and run time due to the feedback loop of these networks.

Future work involves investigating the capabilities of all networks presented here on seasonal data. Additionally, SRNs for medium term wind forecasting will be investigated.

REFERENCES

[1] A. Fabbri, T. Roman, J. Abbad, V. Quezada, "Assessment of the Cost Associated With Wind Generation Prediction Errors in a Liberalized Electricity Market", *IEEE Transactions on Power Systems*, August 2005, pp. 1440-1446.

[2] M. Hable, C. Meisenbach, G. Winkler, "Economically Optimised Power Dispatch in Local Systems Using Evolutionary Algorithms and Dynamic Programming", *Power System Management and Control Conference*, April 2002, pp. 174-179.

[3] National Renewable Energy Laboratory's National Wind Technology Center (online): http://www.nrel.gov/wind/

[4] S. Sanz, A. Perez-Bellido, E. Ortiz-Garcia, A. Portilla-Figueras, L. Prieto, D. Paredes, F. Correoso, "Short-term Wind Speed Prediction by Hybridizing Global and Mesoscale Forecasting Models with Artificial Neural Networks", *Eighth International Conference on Hybrid Intelligent Systems*, Sept. 2008, pp. 608-612.

[5] H. Shuang, L. Yongqian, Y. Yongping, "Taboo Search Algorithm Based ANN Model for Wind Speed Prediction", *2nd IEEE Conference on Industrial Electronics and Applications*, May 2007, pp. 2599-2602.

[6] S. Li, D. Wunsch, E. O'Hair, M. Giesselmann, "Using Neural Networks to Estimate Wind Turbine Power Generation", *IEEE Transactions on Energy Conversion*, Sept. 2001, pp. 276-282.

[7] T. Senjyu, A. Yona, N. Urasaki, T. Funabashi, "Application of Recurrent Neural Network to Long-Term Ahead Generating Power Forecasting for Wind Power Generator", *IEEE PES Power Systems Conference and Exposition*, Oct-Nov 2006, pp.1260-1265.

[8] S. Li, "Wind Power Prediction Using Recurrent Multilayer Perceptron Neural Networks", *IEEE Power Engineering Society General Meeting*, Volume 4, July 2003, pp.13-17.

[9] G. K. Venayagamoorthy, "A Successful Interdisciplinary Course on Computational Intelligence", *IEEE Computational Intelligence Magazine – A special issue on Educational Issues on Computational Intelligence*, Feb. 2009, pp. 14-23.

[10] E. J. Teoh, K. C. Tan, C. Xiang, "Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Decomposition", *IEEE Transactions on Neural Networks*, Volume 17, No. 6, Nov. 2006, pp. 1623-1629.

[11] J. Kennedy, R. Eberhart, *Swarm Intelligence*, Morgan Kauffman Publishers, San Franscisco, CA, 2001.

[12] V. G. Gudise, G. K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks", *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, April 2003, pp. 110-117.

[13] S. Doctor, G. K. Venayagamoorthy, "Optimal PSO for Collective Robotic Search Applications", *IEEE Congress on Evolutionary Computation*, June 2004, pp. 1390-1395.