2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Specific Humidity Forecasting using Recurrent Neural Network

Chen Fang, Xipeng Wang, Yi L Murphey
University of Michigan-Dearborn
Dearborn, MI, USA
yilu@umich.edu

David Weber, Perry MacNeille
Ford Motor Company
Dearborn, MI, USA
{dweber, pmacneil}@ford.com

***Abstract* — This paper presents our research in building a virtual humidity sensor using recurrent Neural Networks. Recurrent Neural Networks are promising methods for the prediction of time series because they provide feedback connections from hidden layer to its inputs and, therefore, can store temporal information learned from previous time steps. This study applies Elman Recurrent Neural Network (ERNN) to forecast the specific humidity from three weather stations. In addition, this study examines the feasibility of applying ERNN in time series forecasting by comparing it with multilayer perceptron network. The experiment results indicate that ERNN is a promising alternative to specific humidity forecasting.**

Keywords — humidity sensor, recurrent Neural Networks

I. INTRODUCTION

Specific humidity (SH), defined as the ratio of the mass of water vapor to the mass of dry air (gm/kg), is an important input for critical vehicle control systems that operate the engine, transmission, climate control and others. Specifically, it is needed to control spark timing, EGR (exhaust gas recirculation) rate, spark advance and climate control functions on vehicles that meet future emissions and fuel economy targets. Atmospheric temperature and SH changes have significant effects on the specific power output, specific fuel consumption and the emissions of nitric oxide and smoke from an automotive diesel (compression ignition) engine [1]. Better use of SH measurements to improve the vehicle engine emission control system is a current area of research [2][3]; as is automatic windshield defogging control system [4], and air conditioning system [5].

A few research works have been conducting in predicting specific humidity. Cheng et al [6] presented a neural network which produces an output based on a linear combination of non-linear physical signals generated by conventional physical sensors. MacNeille et al [7] developed a system for providing weather conditions, which uses a server to communicate with a weather data provision service. The authors believe this paper is the first demonstration of a virtual SH sensor using a neural net trained with historic weather data to predict SH from normal vehicle sensors.

Neural networks offer an alternative approach to the hardware humidity sensors. Prediction of ambient temperature, thermal condition and SH have been studied by researchers in much larger scales, such as global climate modeling [8], building architecture [9][10][11][12], and agriculture [13]. Mathematical models based on auto-regression [10][11][12], artificial neural network [8][14], and numerical estimation [9][15] have been used to address these large scale problems. In [10] a neural network based nonlinear autoregressive model with external inputs (NNARX) was used to predict the thermal behavior of an open office in a modern building. External and internal climate data was recorded over three months then used to build and validate dry-bulb temperature and RH prediction models on several time scales (3 hour and 30 minutes). The accuracy of the multiple step-ahead predictions were tested against multiple performance measures such as goodness of fit, mean squared error, mean absolute error and coefficient of determination between predicted model output and accurate physical measurements. The optimal brain surgeon strategy was used to prune the network structure of the NNARX model to fully optimize the network. The results demonstrate that both the NNARX and numerical estimation models provide acceptable predictions, but the nonlinear NNARX model outperforms the linear ARX model.

In [11], Yigit and Ertunc use a neural network system to predict air temperature and SH at the outlet of a wire-on-tube heat exchanger. A feed-forward neural network is trained to model the thermal performance of the coil using nine inputs: temperature and SH of the air entering the coil, air velocity in the coil, frost weight of the condensation in the coil, the temperature at the coil surface, mass flow rate of the heat transfer fluid, its temperature at the inlet and outlet of the coil and the ambient temperature. The predicted temperature and SH of the air leaving the coil are output by the neural net. The predicted values are found to be in good agreement with the measured values in the tests, with mean relative errors less than 1% for outlet air temperature and 2% for outlet SH.

In contrast to the traditional feed-forward neural networks, recurrent neural network is capable of implementing temporal dynamics, namely representing time parameters implicitly. Its recurrent connections are based precedence relationships while the forward connections are based on dependency relationships [16]. Various RNNs with different architectures and topologies have been studied by researchers. For example, fully recurrent network, Hopfield network, Elman and Jordan network, echo state network, bi-directional network, continuous-time recurrent network, hierarchical recurrent

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

network, recurrent multilayer perceptron and Pollack's sequential cascaded network can all be classified as recurrent neural network [18][19][20][21][22].

Among them, the most basic and popular one may be the Elman network employed by Jeff Elman. A three-layer network is used, with the addition of a set of "context units". There are connections from the middle (hidden) layer to these context units fixed with a weight of one [17]. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units. Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that is beyond the power of a standard multilayer perceptron.

Another famous type of recurrent neural network is Hopfield network. The Hopfield network is of historic interest although it is not a general RNN, as it is not designed to process sequences of patterns. Instead it requires stationary inputs. It is a RNN in which all connections are symmetric. Invented by John Hopfield in 1982 [18], it guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration. A variation on the Hopfield network is the bidirectional associative memory (BAM) [19]. The BAM has two layers, either of which can be driven as an input, to recall an association and produce an output on the other layer.

To accurately forecast the specific humidity, the Elman recurrent neural network is exploited in this paper. The paper is organized as follows: In section II, the implementation algorithms for MLP and ERNN are introduced; in section III, a specific humidity forecasting system is developed using both MLP and ERNN algorithms, and the experiment performance is analyzed; finally in section IV, conclusions are made based on case study.

II. Modeling Humidity sensor using Neural Networks

In this section, we first describe a virtual humidity sensor model, the two neural network systems we use to implement a virtual humidity sensor.

A. Virtual Sensor System for Predicting Ambient Humidity

National Oceanic and Atmospheric Administration (NOAA) provides local climate data for all states in USA (http://www.ncdc.noaa.gov/most-popular-data#loc-clim)

Climate data include outlook, temperature, pressure, dew-point, relative humidity, precipitation, sunrise time, sunset time, wind speed, wind direction, etc. The problem we are trying to solve is to predict specific humidity (SH) using other available weather data.

SH is a function of air-mass, temperature and barometric pressure. The premise of the proposed virtual sensor is that an intelligent system such as a neural network can be trained to determine the SH from the air mass that is influencing the area surrounding the vehicle using the time, date, temperature and barometric pressure, and other signals obtainable from the NOAA.

Mathematically, we describe the problem as follows. At any time t, we intend to predict specific humidity value using a computer system F such that SH(t) = F($\bar{x}$ (t)), where $\bar{x}$ (t) is a feature vector extracted from the available weather at time t, and F is the virtual sensor system that outputs the specific humidity at time t.

Selection of input signals from weather station and extracting effective features for building the virtual sensor is an essential aspect of conceptualizing a virtual sensor. We select the signals that are available at a NOAA weather station. Through scientific analysis we extracted the following signals, day index, hour index, Sun Angle, temperature, barometric pressure, and rainy status. From these signals we extracted 19 features, which are illustrated in Table 1. Among them, sun angle was calculated based on location, time, and date under the assumption that earth is a perfect sphere. Several statistical features were extracted from the raw temperature and barometric pressure signals considering the fact that they are two key parameters in determining specific humidity according to the equation (10). A feature "Rainy Status" was decoded from the CAN bus signal "wiper status" in a vehicle. The "rainy" situation includes 7 possible weather types, which are rain, snow, snow grains, ice crystals, ice pellets, hail and small hail. Several other raw signals, such as wind speed, wind direction, and visibility were not considered for feature extraction because they are not available in current vehicle CAN bus system.

Below features serve as input to the neural works described below.

TABLE I. FEATURE LIST OF LOCAL AMBIENT HUMIDITY ESTIMATION SYSTEM

Index	Symbol	Description
1	DD_t	Day index (1 - 31)
2	HH_t	Hour Index (0-23)
3	S_t	Sun Angle (0 – 90 degree)
4/5/6	T_t / T_{t-1} /	Temperature at current time/5-minute ago/10-minute ago
7	T_t^{av1}	Mean Temperature of past 1 hour
8	T_t^{av6}	Mean Temperature of past 6 hours
9	T_t^{av24}	Mean Temperature of past 24 hours
10	T_t^{std1}	Temperature Std. of past 1 hour
11	$T_t^{\min 1}$	Min Temperature of past 1 hour
12	$T_t^{\max 1}$	Max Temperature of past 1 hour
13/14/15	P_t / P_{t-1} / P_{t-2}	Barometric Pressure at current time/5-minute ago/10-minute ago
16	P_t^{std1}	Barometric Pressure Std. of past 1 hour
17	$P_t^{\min 1}$	Min Barometric Pressure of past 1 hour
18	$P_t^{\max 1}$	Max Barometric Pressure of past 1 hour
19	RS_t	Rainy Status (0-nonrainy, 1-rainy)

B. Multilayer Perceptron

The most popular neural network architecture in use today is perhaps the multi-layer perceptron for both classification and regression [20]. The basic MLP equation is given as follows,

$$y = \upsilon_0 + \sum_{j=1}^{NH} \upsilon_j sig(\omega_j^T \, x') \quad (1)$$

where NH represents the number of hidden nodes, x is the input vector, augmented with 1, i.e., $x' = (1, x^T)^T$. ω_j is the weighs vector of jth hidden node, υ_j is the weighs vector for jth output node, and y is the neural network output. The function sig represents the activation function of hidden nodes, namely the log-sigmoid function in formula (2)

$$\mathrm{sig}(\mu) = \frac{1}{1+\exp(-\mu)} \quad (2)$$

As a heavily parameterized model, MLP's complexity largely depends on the number of hidden nodes NH. The complexity of MLP can be flexibly controlled by selection proper number of hidden nodes. The universal approximation property is perhaps the greatest breakthrough that lent credence to the capacity of neural network [21][22][23][24]. Under certain mild conditions on the hidden nodes activation function, any given continuous function on a compact set can be approximated as close as arbitrarily given using a network with a finite number of hidden nodes. For this reason, it important to avoid over-parameterization, especially when large amount of noise is contained in input data during forecasting. K-fold validation procedure is usually employed to select the proper number of hidden nodes.

During training process, the most well-known learning algorithm, based on the steepest descent concept, is the back-propagation algorithm. Another famous second order optimization method called Levenberg Marquardt is usually regarded to be more efficient than the basic back-propagation algorithm [25]. Also this second order algorithm is used in our experiment implementation. (Matlab function trainlm)

C. Elman Recurrent Neural Network

Elman Recurrent Neural Network is one of the most famous recurrent neural networks proposed by Elman in 1990 [17]. It utilizes the well-known back-propagation training algorithm and feedbacks connections from the hidden layer to its inputs. Therefore the input of the recurrent hidden layer consists of two parts: the true input from the input data, and the context input, a copy of the activations of the hidden layer from previous time step. For this reason, the Elman recurrent neural network is able to store the temporal information back to any previous states. Different from the feed-forward connections, the weights of feedback recurrent connections are usually fixed as 1 and are not subject to weights updating during training.

Sigmoid function serves as the activation function for the hidden layer and the activation function for the output layer is linear. The number of output nodes is denoted by S_2, number of hidden nodes by S_1, input nodes by R. The architecture of Elman recurrent neural network is shown as figure 1. At any time step t, the output of j_{th} hidden node is further denoted as $V_j(t)$ and can be computed using formula (3),

$$V_j(t) = \mathrm{sig}[\sum_{k=1}^{R} \omega_{jk} I_k(t) + \sum_{k=R+1}^{R+S_1} \omega_{jk} V_k(t-1)] \quad (3)$$

, where $V_j(t-1)$ is the output of j^{th} hidden node at time t-1, $I_k(t)$ the k_{th} input at time t, and w_{jk} the connection weight between the k_{th} input node (including the both true inputs and context inputs) and the j_{th} hidden node. Also the connection weighs between context input nodes and hidden nodes are 1 shown as formula (4)

$$\omega_{jk} = 1, \quad when \quad k = R+1, R+2, \ldots, R+S_1 \quad (4)$$

Let the output of i_{th} ouput node be denoted as $O_i(t)$, and the connection strength between j_{th} hidden node and i_{th} ouput node as W_{ij}, then the linear activation function for the output layer is illustrated in formula (5).

$$O_i(t) = \sum_{j=1}^{S_1} W_{ij} V_j(t), \quad i = 1, \ldots, S_2 \quad (5)$$

The back-propagation learning algorithm is utilized to train this Elman recurrent neural network. The connection weights between node q and p at a learning iteration step l are updated according to following formula (6)

$$\Delta\omega_{pq}^m(l) = (1-\alpha)\eta \sum_{t=1}^{N} \delta_p^m(t) \, V_q^{m-1}(t) + \alpha \Delta\omega_{pq}^m(l-1), \quad (6)$$

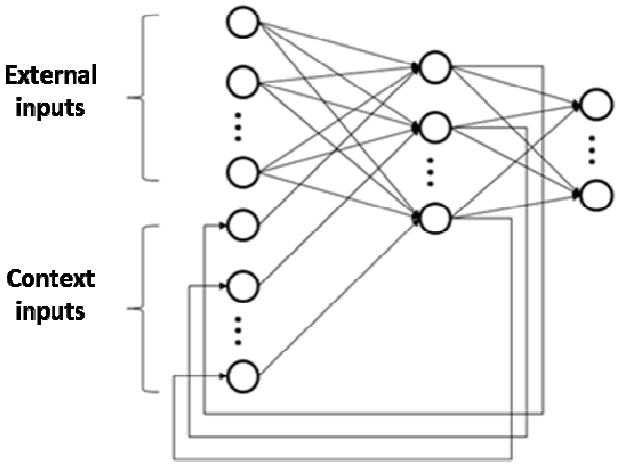

Fig.1. Architecture of Elman Recurrent Neural Network

where η is the learning rate, and the α represents the momentum parameter, m the number of layers of network, N the number of input data samples. In the two-layer Elman recurrent neural network example, hidden layer is represent by m=1, output layer by m=2. δ can be computed using formula (7) and (8)

$$\delta_i^{m=2}(\mathrm{t}) = T_i(\mathrm{t}) - O_i(\mathrm{t}), \quad i = 1, 2, \ldots, S_2 \tag{7}$$

$$\delta_j^{m=1}(\mathrm{t}) = [1-(\mathrm{V}_j(\mathrm{t}))^2]\sum_i W_{ij}^{m=2}\delta_i^{m=2}, \quad j = 1, 2, \ldots, S_1 \tag{8}$$

, where $T_i(t)$ is the prediction true target and $O_i(t)$ is the neural network output. The momentum parameter α and learning rate η will be automatically adjusted and obtained during training process based on the Hertz adaptive learning algorithm [26] shown by formula (9)

$$\Delta\eta = \begin{cases} 0, & otherwise; \\ -a\eta, & if \quad \Delta E > 0; \\ +b\eta, & if \quad \Delta E < 0. \end{cases} \tag{9}$$

, where the ΔE is the cost function change, and a, b are constant positives.

D. A system of Neural Networks for Predicting Humidity Value

For predicting humidity values, we train one neural network for each month of the year. Each month, a neural network is trained. NN_i represents the neural network trained using the features shown in Table 1. The training data is historical weather data collected for the month before and after i, and the ith month of the same year period. Figure 2 illustrates the data used to train a neural network to predict humidity measures in the month of January.

III. EXPERIMENTS

In this section, three neural network systems were developed to estimate the ambient specific humidity in three USA cities, Detroit, Houston, and Phoenix, and three systems were further evaluated using year 2012's testing data. There are basically two objectives in performing ambient specific humidity estimation using downloaded weather station data.

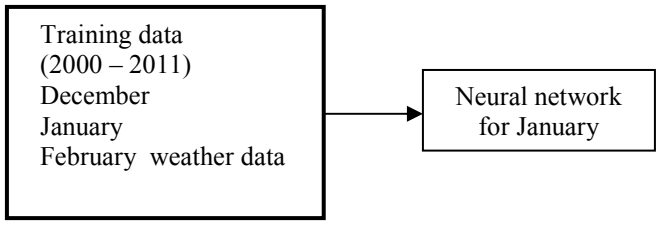

Fig.2. Illustration of data used to train a humidity sensor.

The primary objective is to prove the concept that an in-vehicle physical humidity sensor can be replaced with a software module. In another word, with free weather station data in hand, we design experiments to test the feasibility of estimating the specific humidity signal using only signals that are available in a vehicle's CAN (Controller Area Network) bus system. The secondary objective is that the accurately estimated local ambient specific humidity can be further employed as an input signal to improve the performance of the vehicle intake air specific humidity estimation system, considering the fact that vehicle intake air specific humidity is usually highly correlated with local ambient specific humidity.

A brief introduction about the data source is summarized in the "Data Collection" subsection; the details about the estimation system's input features, structure, and algorithms are included in the "Method" subsection. In the last subsection, we present the experiment design and results.

A. Data Collection

The weather data from three local weather stations, Detroit, Houston, and Phoenix, were collected from NOAA official website [27]. The time span and resolution of the dataset is shown in Table 2. A variety of signals, such as station number, date, visibility, temperature, barometric pressure, wind speed, et al, are included in the raw dataset. Signal preprocessing was performed to handle the data missing and invalidity problem due to the reported station sensor malfunctions during certain periods.

B. Specific Humidity Calculation and System Performance Metric

Both of the downloaded weather station data and collected real vehicle data contain relative humidity signal instead of specific humidity signal , so an equation is used to calculate specific humidity at every data sample given relative humidity, temperature and barometric pressure.

$$SH_t = \frac{3801 \times RH_t \times \exp(\frac{17.62 \times T_t}{243.5 + T_t})}{P_t - 6.112 \times RH_t \times \exp(\frac{17.62 \times T_t}{243.5 + T_t})} \tag{10}$$

The formula (10) illustrates the conversion equation, where the SH_t (g/Kg) is the calculated specific humidity at time t, the RH_t (g/Kg) is the relative humidity at time t, T_t (Celsius) is the ambient temperature at time t, and the P_t (hPa) is the barometric pressure at time t. Contact authors for 'Ford Engineering Specification for a Physical Specific Humidity Sensor' for the raw conversion formula and detailed explanation. (Document is not allowed to be disclosed due to the confidential clause)

$$Error \quad Rate = \frac{Number \quad of \quad out-of-bound \quad estimates}{Number \quad of \quad all \quad estimates} \tag{11}$$

$$Accuracy = 1 - Error \quad Rate \tag{12}$$

TABLE II. DATA SUMMARY OF LOCAL AMBIENT HUMIDITY ESTIMATION SYSTEM

Location	Time Span	Resolution
Detroit(DTW)	Jan 2000 to Feb 2013	5 minutes
Houston(HOU)	Mar 2005 to Feb 2013	5 minutes
Phoenix(PHX)	Jan 2000 to Feb 2013	5 minutes

The formula (11) and (12) illustrate the definition of system "error rate" and "accuracy" respectively. If the absolute value of the difference between a system estimated specific humidity and the truth value is greater than 2.5g/Kg, then this estimate is defined as an "out-of-bound" estimate. This accuracy concept instead of the 'mean square error' is used to evaluate system performance in this paper because it is consistent with the current auto industry development specification. Contact authors for 'Ford Engineering Specification for a Physical Specific Humidity Sensor' for more details. (Document is not allowed to be disclosed due to the confidential clause)

C. *Training and Testing Data Partition Strategy*

A humidity detection system consists of 12 neural networks to estimate ambient specific humidity, one for each month of the year. The systems were trained using the weather data recorded every 5 minutes between 2000 and 2011 for two cities, Detroit and Phoenix, between 2005 and 2011 for one city, Houston. The systems were evaluated using the weather data recorded in the year of 2012. Also, to train and test a neural network for one city and one month, only historical adjacent months' data for that city were used as training data. For example, when developing a neural network for December of Detroit, we would train the system using November, December, and January's data between 2000 and 2011 in Detroit as the training data, and test the system using 2012 December's data in Detroit.

D. *Experiment Design and Results*

For MLP algorithm, different numbers of hidden nodes, 5, 10, 15, and 20, were experimented. Epoch and learning rate were set as 5000 and 0.1 respectively after trials and errors. For ERNN algorithm, different numbers of hidden nodes, 5 and 10, were experimented. Epoch, learning, and delayed feedback steps were set as 5000, 0.1, and 2 respectively after trials and errors. The system performance is illustrated in Table 3. The algorithm by using which system produced greater testing accuracy was deemed as optimal one, and was therefore displayed under the "optimal algorithm" column. In estimating the ambient specific humidity in the Detroit area, with the 11 months of available test data, the system reached 2-sigma accuracy requirement (95.4%) for nine months, and 3-sigma accuracy requirements (99.7%) for seven months. In the Houston and Phoenix area, with the 12 months weather data in 2012, the system reached 2-sigma accuracy requirement for eleven and ten months respectively, and 3-sigma accuracy requirements for four and two months respectively.

Figures 3 to 5 show 3 examples of ERNN prediction example.

TABLE III. PERFORMANCE SUMMARY OF LOCAL AMBIENT HUMIDITY ESTIMATION SYSTEM

	Houston		Detroit		Phoenix	
	Testing Accur.	***Optimal Algor.***	***Testing Accur.***	***Optimal Algor.***	***Testing Accur.***	***Optimal Algor.***
Jan.	97.47%	ERNN	100%	ERNN	99.17%	ERNN
Feb.	98.18%	ERNN	100%	ERNN	100%	ERNN
Mar.	97.09%	ERNN	99.83%	MLN	97.90%	ERNN
Apr.	97.60%	ERNN	100%	ERNN	98.67%	ERNN
May	99.47%	MLN	93.42%	ERNN	100%	MLN
Jun.	99.84%	MLN	91.88%	MLN	95.97%	ERNN
Jul.	100%	MLN	NA	NA	73.93%	MLN
Aug.	100%	MLN	97.35%	MLN	97.76%	MLN
Sep.	97.18%	MLN	95.49%	ERNN	93.39%	MLN
Oct.	92.26%	MLN	100%	ERNN	96.37%	ERNN
Nov.	96.84%	ERNN	100%	ERNN	96.11%	ERNN
Dec.	100%	ERNN	100%	ERNN	97.89%	ERNN

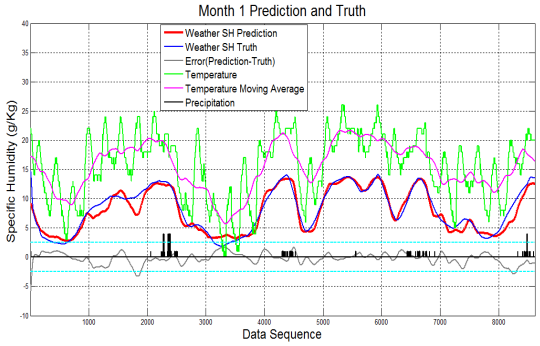

Fig.3. ERNN prediction performance for Houston January data

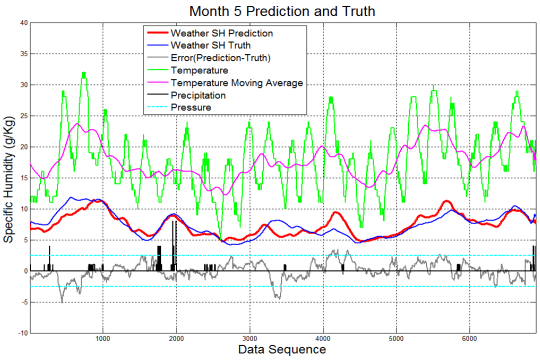

Fig.4. ERNN prediction performances for Detroit May data

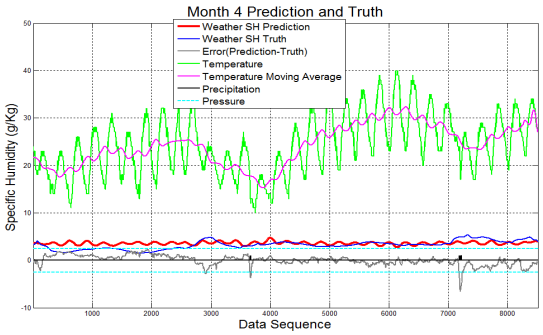

Fig.5. ERNN prediction performances for Phoenix April data

IV. CONCLUSION REMARKS

In this paper, MLP and ERNN algorithms are studied and utilized respectively to develop a weather station specific humidity forecasting system. We evaluated the neural network architectures using the weather data downloaded from the web. The results of our experiments show that the ERNN algorithm has great potential and can generally beat MLP during winter time when the data noise is relatively small. On the other hand, ERNN may not perform as good as a MLP when larger amount of noise is contained in the training and test data, since errors are accumulated through recurrent feed-back connections. Other state-of-art time series forecasting methods will be further studied and experimented in the future work.

REFERENCES

[1] Rakopoulos, C., "Influence of ambient temperature and humidity on the performance and emissions of nitric oxide and smoke of high speed diesel engines in the Athens/Greece region," Energy Conversion and Management, Vol 31, Issue 5, Pages 447-458, 1991

[2] Liang, C., Srinivasan, S. and Jacobson, E., "NOx Emission Control System Using a Virtual Sensor," United States Patent, No. US6882929B2, 2005

[3] Kubesh, J., Podnar, D., Latusek, J. and McCaw, D., "Engine Control to Reduce the Emission Variability," United States Patent, No. US6581571B2, 2003

[4] Niimi, N., Yoshida, T. and Isogai, T., "Capacitive Humidity Sensors Using Highly Durable Polyimide Membrane," SAE Int. J. Passeng. Cars – Electron. Electr. Syst. 6(1):2013, doi:10.4271/2013-01-1337.

[5] Wu, X., Johnson, P. and Akbarzadeh, A., "Application of heat pipe heat exchangers to humidity control in air conditioning systems," Applied Thermal Engineering, Vol 17, Issue 6, Pages 561-568, 1997

[6] Cheng et al., "Virtual Vehicle Sensors Based On Neural Networks Trained Using Data Generated By Simulation Models," U.S. Patent 6,236,908, May 22, 2001.

[7] MacNeille et al., "Virtual ambient weather condition sensing," U.S. Patent Application US20120158207, June 21, 2012.

[8] Rehman, S. and Mohandes, M., "Artificial neural network estimation of global solar radiation using air temperature and relative humidity," Energy Policy 36 (2008) 571–576, 2008

[9] Sloane, S. and Wolff, T., "Prediction of ambient light scattering using a physical model responsive to relative humidity: validation with measurements from Detroit," Atmospheric Environment Vol. 19,No. 4, pp.669480,1985

[10] Mustafaraj, G., Lowry, G. and Chen, J., "Prediction of room temperature and relative humidity by autoregressive linear and non-linear neural network models for an open office," Energy and Buildings 43 (2011) 1452–1460, 2011

[11] Yigit, K. and Ertunc H., "Prediction of the air temperature and humidity at the outlet of a cooling coil using neural networks, " International Communications in Heat and Mass Transfer 33 (2006) 898–907, 2006

[12] Sigumonrong, A., Bong, T., Fok, S. and Wong, Y., "Self-learning Neurocontroller for Maintaining Indoor Relative Humidity," International Symposium on Neural Networks-ISNN, vol. 2, pp. 1297-1301 vol.2, 2001

[13] Peter E Thornton, Hubert Hasenauer, Michael A White, Simultaneous estimation of daily solar radiation and humidity from observed temperature and precipitation: an application over complex terrain in Austria, Agricultural and Forest Meteorology, Volume 104, Issue 4, 15 September 2000, Pages 255-271, ISSN 0168-1923, http://dx.doi.org/10.1016/S0168-1923(00)00170-2.

[14] Lu, T. and Viljanen, M., "Prediction of indoor temperature and relative humidity using neural network models: model comparison," Neural Comput & Applic (2009) 18:345–357, 2009, doi: 10.1007/s00521-008-0185-3

[15] Teodosiu, C., Hohota, R, Rusaouen, G. and Woloszyn, M., "Numerical prediction of indoor air humidity and its effects on indoor environment," Building and Environment 38 (2003) 655 – 664, 2003

[16] Fu, L. M., "Neural Networks in Computer Intelligence," ch. 10. McGraw-Hill, Inc., 1994

[17] H. Cruse, "Neural Networks as Cybernetic Systems," 2nd and revised edition, 2009

[18] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings National Academic Science USA. 1982 April; 79(8): 2554–2558.

[19] R. Rojas, "Neural networks: a systematic introduction. Springer," Springer-Verlag, Berlin, New-York, 1996

[20] Bishop, C. M., "Neural Network for Pattern Recognition," UK, Oxford University Press, 1995

[21] Cybenko, G., "Approximation by Superposition of Sigmoidal Functions," Mathematics of Control, Signals and Systems, 2:303-314, 1989

[22] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," Neural Networks, 2:183-192, 1989

[23] Hornik, K. Stinchombe, M., White, H., "Multi-layer Feedforward Networks are Universal Approximators," Neural Networks, 2:359-366, 1989

[24] Leshno, M., Lin, V., Pinkus, A., Schocken, S., "Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function," Neural Networks, 6:861-867, 1993

[25] Jorge J, More, "The Levenbery-Marquardt algorithm: Implementation and Theory," Proceedings of the Biennial Conference, Dundee, June, 1977

[26] Hertz, J., A. Krogh and R. G. Palmer, "Introduction to the Theory of Neural Computation," ch. 6, Addison-Wesley, 1991

[27] http://www.ncdc.noaa.gov/